PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30 — A1

(11) International Publication Number: **WO 97/15018**

(43) International Publication Date: 24 April 1997 (24.04.97)

(21) International Application Number: PCT/US96/15620

(22) International Filing Date: 26 September 1996 (26.09.96)

(30) Priority Data:
08/543,644 16 October 1995 (16.10.95) US

(71) Applicant: BELL COMMUNICATIONS RESEARCH, INC. [US/US]; 445 South Street, Morristown, NJ 07960-6438 (US).

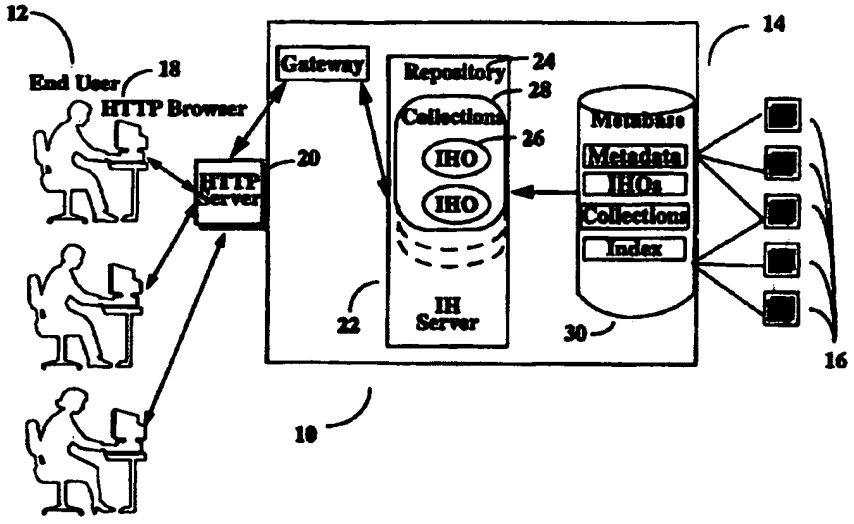
(72) Inventors: MARCUS, Howard; 12 Harrison Street, Edison, NJ 08817 (US). SHAH, Kshitij, Jawahar; 119 Jeremy Court, Edison, NJ 08817 (US). SHETH, Amit, Pravinkumar; 1140 Laurel Pointe, Bogart, GA 30622-2856 (US). SHKLAR, Leon, A.; 160 Stults Lane, East Brunswick, NJ 08816 (US). SURAK, Jerome, Raymond; 277 Mohawk Trail, Bridgewater, NJ 08807 (US). THATTE, Satish, Mukund; 18 Crestview Drive, Kendall Park, NJ 08824 (US).

(74) Agents: WHITE, Lionel, N. et al.; International Coordinator, Room 1G112R, 445 South Street, Morristown, NJ 07960-6438 (US).

(81) Designated States: CA, CN, JP, KR, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: METHOD AND SYSTEM FOR PROVIDING UNIFORM ACCESS TO HETEROGENEOUS INFORMATION



(57) Abstract

Our invention is a system and methodology for integrating heterogeneous information in a distributed environment by encapsulating data about existing and new information into objects (16). The process of encapsulating the information requires extracting from the information metadata. Creating from the metadata, a database (30), where the metadata is grouped into objects (26) and groups of objects (28) which are logically associated into collections (28). This database of object and collections is instantiated into runtime memory of a server (22), organized into repositories (24) of objects (20) and collections (28). A user (12) seeking access to the information would then, using an HTTP compliant browser (20), access the server (22) to access the information through the objects (26) created and stored in the server.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

PLATFORM-INDEPENDENT UNIVERSAL DATA ACCESS SYSTEM AND METHOD IN A CLIENT-SERVER ENVIRONMENT

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to client-server software systems and particularly to data access in a client-server environment.

2. Description of the Related Art

In the 1960's and 1970's computing was performed primarily by large mainframe computers servicing multiple users. Most users accessed the mainframe computer using a "dumb" terminal that had very little memory and computing power other than what was required to interface to the mainframe and to present data received from the mainframe computer to the user. In this scenario, each user was given time shared access to the computing cycles of the mainframe as well as to the programs and data stored centrally on the mainframe.

With the development and proliferation of the personal computer in the 1980's another computing paradigm took hold. In this paradigm, a user worked on a dedicated "personal" computer and accessed a private set of application programs and data stored on the personal computer. Further, with this paradigm, no time sharing of processing cycles was necessary because the personal computer served the single user only.

Another change in the computing paradigm was caused by the development of computer networking technology. Computer networking

technology links multiple computers to enable the transfer of data. The development of computer networking technology was particularly important in the corporate environment where sharing and exchanging information is of particular importance. One network architecture that has developed is the client-server architecture. In this architecture, one or more computers are configured as a "server." A server stores a set of shared resources such as data (generally organized by storage within the context of a database) and applications. Further in accordance with this architecture, one or more other computers are configured as clients to access the stored data and applications from one or more servers.

Using this client-server architecture many companies have set up a system where various categories of corporate data are stored on various servers. In many cases, the client computers are based on different hardware platforms and run different operating systems. Further, in many cases, the server computers are just as varied in terms of hardware and operating system environments. For example, it is typical for a company to store human resources information in one type of database, inventory information in another, engineering documents and so on. Further, companies often need to access data stored in other types of data sources (storage vehicles) such as in an electronic mail system, or from a large mainframe data source.

Accessing and/or updating data stored in these varied storage systems requires using a data management system tailored for accessing the particular type of stored data or directly invoking the application which stores the data (e.g. an email program, a word processor, the specific database and the like). Thus a user must learn separate interfaces for a variety of software applications in order to access, view and modify data from multiple sources. Additional system complexity is introduced when data access is desired from

a variety of computer platforms operating different operating system types. In these cases, conventional data access system and methods are ported for compatibility with more than one type of computer.

Thus, there is a need for a system and method for accessing data that is stored in a variety of databases, in a variety of formats and on a variety of different types of computers.

SUMMARY OF THE INVENTION

In accordance with the present invention, a computer implemented system and method provides platform independent access to data stored in any one of a variety of data storage mediums such as SQL databases, mainframe databases, mass storage mediums, mediums accessible via a driver, electronic mail systems and the like. The invention includes a client module that provides a desktop operating environment having a desktop window operating "inside" a browser. The desktop window presents one or more books each representing data stored in a corresponding storage medium (e.g., on a particular server computer) and in one or more particular storage formats (e.g., in an SQL database format or in a particular electronic mail format). The books are represented by icons having a user-customized name. The invention also includes a server module in communication with multiple client modules. The server module includes a metadata for accessing data associated with each book. The server module further includes computer instructions that process data requests (access requests) received from the client module by (i) retrieving the metadata associated with the book, (ii) retrieving data corresponding to the metadata; and (iii) sending the retrieved data to the client module.

In accordance with the present invention, the client module is a browserware application. A browserware application is program code that is stored on a server and is accessed from and run within a browser running on a client computer. Advantageously, a browserware application (i) does not require user installation on the client side, (ii) is hardware and operating system independent on the client side and (iii) allocates a reduced amount of processing to the client computer thus reducing the use of memory and processing resources on the client computer.

Further in accordance with the present invention, at a book provided by the client module is linked to (assocaited with) to data stored in a data source and to program code for viewing the stored data. Further, information stored in the server module's data object associates the name associated with the book with physical storage information (e.g., metadata) for retrieving the data. Data is then retrieved by processing the physical storage information to determine the location and format of the data being accessed. If the data resides in a database, then the location of the database is retrieved from the data object and an appropriate query is constructed then applied to the database.

Still further in accordance with the present invention, the client module runs in a browser environment. The browser environment manages the interaction between the client module and the client computer operating system. The client module thus does not require installation on or configuration with the operating system of the client computer. Preferably the browser environment is a Java-compliant browser environment and the client module is a Java-applet.

Yet further in accordance with the present invention, retrieved data is presented to a user in accordance with a viewing form and the system

additionally includes a scripting module for generating a customized viewing form. The scripting module includes a byte-code generator disposed to generate byte-code in accordance with user specified form layout. Advantageously, because the scripting module generates browser-compatible byte-code, the customized form displays data to the user from within the browser environment independent of the computer hardware platform or operating system on which the browser is running.

Yet still further in accordance with the invention, the invention includes a user interface for configuring the server module for accessing data stored in any one of a plurality of storage mediums in any one of a plurality of storage formats. The user interface allows a user to specify physical storage information associated with data stored in one or more storage vehicles and stored in one or more data formats. The invention also includes a processing module in communication with the user interface. The processing module generates a data object using the information received from the user via the user interface. The data object associates the received physical storage information with a name (e.g. a data identifier).

The features and advantages described in the specification are not all-inclusive, and particularly, many additional features and advantages will be apparent to one of ordinary skill in the art in view of the drawings, specification, and claims hereof. Moreover, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter, resort to the claims being necessary to determine such inventive subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a functional block diagram of a data access system in accordance with the present invention.

Figure 2 is a diagram of the data organization structure of a book provided by the data access system shown in Figure 1.

Figure 3 is a flow diagram of object to relational mapping to access data in accordance with the data access system shown in Figure 1.

Figure 4 illustrates a desktop window of the desktop client module shown in Figure 1.

Figure 5 illustrates a new manager dialog window provided by the data access system shown in Figure 1.

Figures 6A-6C illustrate new book dialog windows provided by the data access system shown in Figure 1.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The Figures depict a preferred embodiment of the present invention for purposes of illustration only. One skilled in the art will readily recognize from the following discussion that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the invention described herein.

Figure 1 is a functional block diagram of a data access system **100** in accordance with the present invention. System **100** includes a client computer **101**, a server computer **102** and data sources **108** coupled via a computer network **103**. Client computer **101** runs a desktop client **105** that provides access to one or more of the data sources included in data sources **108**. Access is provided by a data service application **106** running on a server

computer **102**. Data service application **106** processes data access requests received from desktop client **105** and returns the requested data for presentation to a user from desktop client **105**. Desktop client **105** advantageously runs within a browser **104**, eliminating the need for customization and configuration of desktop client **105** to a particular hardware configuration or to a particular operating system type.

Client computer **101** is a personal computer, a computer workstation of other computer type suitable for running browser **104**. Browser **104** is preferably a conventional Java-enabled (Java-compliant) web browser such as Netscape Navigator 3.0 or Microsoft Explorer 3.0. A Java-enabled browser is a browser adapted to run byte-code (written in the Java programming language) within the browser environment. The Java programming language is an object oriented programming language developed by Sun Microsystems and is defined in the Java Language Specification Version 1.0 by James Gosling, Bill Joy and Guy Steele (published by Addison-Wesley, August 1996) and in the Java Virtual Machine by Tim Lindholm and Frank Yellin (published by Addison-Wesley, September 1996), the contents of which are hereby incorporated by reference. In particular, browser **104** includes a Java Virtual Machine (JVM) that translates Java byte-code into object code for execution. It should be understood that although a preferred embodiment uses a Java-compliant browser, the principles of the present invention apply to uses of other browser types and uses with other types of byte-code virtual machines.

Referring still to Figure 1, data access system **100** includes desktop client **105** and an application programming interface (API) **123**. Desktop client **105** is a "browserware application" that provides a user with access to one or more of the data sources coupled to server computer **102**. A browserware application is program code that is stored on a server and is accessed from and

run within a browser running on a client computer. Advantageously, a browserware application (i) does not require user installation on the client side, (ii) is hardware and operating system independent on the client side and (iii) allocates a reduced amount of processing to the client computer thus reducing the use of memory and processing resources on the client computer.

In a preferred embodiment, desktop client **105** is a set of Java-compatible byte-code instructions forming a Java applet that runs within a Java-compliant browser. Desktop client **105** is identified by a tag. The tag in turn is embedded in a world wide web (Web) page written in hypertext markup language (HTML). Desktop client **105** is accessed (e.g., downloaded to the client computer and run within the browser) when the tag contained a Web page is used to activate a link from within browser **104**.

When desktop client **105** is running, browser **104** manages the interaction between desktop client **105** and client computer **101** (e.g., browser **104** provides an interface between desktop client **105** and client computer **101** and its operating system). Desktop client **105** therefore runs "within" browser **104** rather than interfacing directly with the operating system associated with client computer **101**. The compatibility of desktop client **105** is therefore advantageously determined by compatibility with browser **104** rather than with the type of hardware or operating system associated with client computer **101**. Further, desktop client **105** need not be configured to the particular hardware associated with client computer **101**. Thus, desktop client **105** is platform independent and can run the same set of instructions on a variety of computer hardware and operating systems, provided each is running a browser that includes a virtual machine that can execute the client desktop **105** byte-code. This platform and hardware independence eliminates the need for porting code to achieve multiplatform compatibility. Instead, a

single version of desktop client **105** will execute on any platform having a Java virtual machine.

Server computer **102** is a computer configured to serve multiple client computers **101**. For purposes of simplifying Figure 1, the detail of only a single client computer **101** is shown. Client computer **101** is coupled to server computer **102** using network **103**. Network **103** is preferably an internal TCP/IP intranet although other networks uses such as the Internet or an extranet are used in accordance with the invention.

Server computer **102** runs (executes) a data service application **106**. Data service application **106** provides users at various client computers **101** with access to one or more data sources included in data source set **108** Data service application **106** includes server side API **124**, API proxy **125**, forms module **126**, view module **127**, script module **128**, replication module **129**, security module **130**, script compiler **131**, data access module **132** administration ("admin") tool **133**, data source interface (DSI) **107** and JDBC **110**.

Administration tool **133** is a program module that provides for system configuration and for the selection of data sources within data source set **108**. Further, administration tool **133** allows a user such as a system administrator) to construct, for each selected data source, an access control list (ACL) specifying which users or user groups, are offered access to the particular data source.

Server side API **124** is an application programming interface that processes information communicated to and from various desktop clients **105**. API **124** interfaces with API proxy **125**. API proxy **125** is a program module that implements a remote method invocation (including object serialization) mechanism. The mechanism processes method calls received

from desktop client **105** for remote invocation. Remote invocation means that methods invoked at the client computer **101** side (e.g., by desktop client **105**) actually invoke methods on server computer **102** (e.g., on data service application **106**). By remotely executing methods rather than downloading such methods to client computer **101** for execution on client computer **101**, the use of client computer resources such as memory and processing power is conserved. API proxy **125** thus advantageously reduces use of resources of client computer **101**.

View module **127** is a program module that presents a user with a high level representation of data retrieved from a data source (e.g., a database **113**). In one embodiment, data is presented in a row and column format where, in a design mode, the user selects which fields of a data source are displayed.

Form module **126** is a program module that presents information associated with a selected row of data presented by view module **127**. Selected information is shown in a customized form. Customized forms include widgets. Widgets are visual components such as buttons which, in conjunction with scripts cause the form to modify the presentation of data and perform presentation related processing.

Script **128** is processing module that reads and executes or set of custom scripts (in byte-code) that are associated with the various widgets of a form or view.

Script compiler **131** is a processing module that translates user-generated script text into byte-code.

Replication module **129** is a processing module that provides functionality for mobile access of data. Replication module **129** copies information from a database, provides for its modification and copies it back into the database without performing an over-write.

Security module **130** is a processing module that manages security functions. Security module **130** encrypts data communicated between desktop client **105** and data service application **106**. Security module **130** additionally manages user account and password information.

Data access module **132** is a data access layer that manages information associated with views, forms, scripts, server names, user names, passwords and the like. Data access module **132** interfaces directly with data source interface (DSI) **132**. Data access module **132** translates information between from the format used by API **124** and the format used by DSI **107**.

DSI **107** is a set of computer instructions that provide an interface between data service application **106** and data source set **108**.

Data source set **108** include structured query language (SQL) databases **113**, mail system **112**, native database **121** and mass storage **134**. Mail systems **112** are conventional electronic mail systems including IMAP4 systems **115**, POP3 systems **116**, MAPI systems **117** and VIM systems **118**. Native database **121** is a database native to data access system **100**. Native database **121** provides an alternative data storage source and also stores administrative data used by data service application **106**. Mass storage data source **134** is a storage device for information with large memory needs such as image and video data. SQL databases **113** are structured query language (SQL) databases, each accessed through a server computer **102**. The various data sources included in data source set **108** can each reside on separate computers. It should be understood that the principles of the present invention apply with data sources residing on other computer configurations including where multiple data sources reside on a single computer.

Data service application **106** additionally includes Java database connectivity module (JDBC) **110**. JDBC **110** is a conventional application

programming interface for querying conventional SQL database available from Sun Microsystems. JDBC **110** provides capability to query databases **113** to determine metadata including their connection status (e.g., whether a connection is established) and access, format and syntax information not specifically defined by the SQL.

Figure 2 is a diagram of the data organization structure of a book **200** in accordance with the data access system **100**. Book **200** is named personal contacts and provides access to information associated with personal contacts stored in a database. Book **200** includes a database identifier **201**, a table identifier array **202**, a join condition **203**, and a plurality of views **204**. Database identifier **201** stores information identifying a particular database on a particular server computer. Database identifier **201** is used to construct an address to communicate with the database corresponding to the database identifier. Table identifier **201** is an array of table identifiers that specifies one or more tables in the database. Join condition **203** specifies a particular join function to be used to combine the tables listed by table identifier **201**. Views **204** each specify a custom view of the joined data. View **204** includes a search condition **205**, a selection (list) of attributes **206**, and a plurality of forms **207**. Search condition **205** is the equivalent of a where clause in an SQL statement. Selection of attributes is a list of attributes (fields) to be presented in the view. View **204** provides information that is used by service application **106** to construct a request for a database search result. Service application **106** processes the request, queries the selected database and returns the result set in a dynamically created object.

Figure 3 is a diagram of a dynamically created result set **300**. Result set **300** includes a set of result objects **301**. Result objects **301** are each person objects including a set of attributes **302** (fields) associated with data

corresponding to the person. Attributes **302** includes, for example, start date, fax number, cell number, work number and home number. Each person object **301** is generated as a result of a database query constructed using a JDBC SQL call **303**. The JDBC call **303** performs a table join of the person table **304** and the phone number table **305** associated with the database identified by book **300**. The table join is perform using the person I.D. as the sole join column. Each returned person object is presented within a view as a row. Further customization and refinement of view data is performed using forms **207**. Forms **207** enable selective viewing of data within each view based on column characteristics.

In operation, data access system **100** is preferably first configured by a system administrator for access by multiple users. These multiple users are each preferably identified by user identifications (I.D.s) and by passwords. Users are preferably organized into "groups" based on characteristics tailored to a particular organization's need. For example, a company might arrange users into the following groups: engineers, sales staff, marketing, finance, test and production. The same company might additionally define a different set of user groups such as: part-time employees and full-time employees.

During configuration, a system administrator selects data sources from data source set **108** and for each selected data source, generates an access control list (ACL) identifying which users or user group are given access to the particular data source. Administration tool **133** includes the program instructions that effect such system configuration. Further description of administrative configuration is described in the Admin and DataSource ACL Manager sections of the user's manual included in this specification.

Once data access system **100** is configured as described above, then system **100** provides users with access to selected data sources from desktop

clients **105**. On start up, a desktop client **105** is downloaded to a client computer **101** and service application **106** performs an initialization process in accordance with the system configuration. During the initialization process, service application **106** polls (using JDBC **110** for queuing database sources) each data source in system **100** to determine whether a current connection can be established. Provided that a current connection is established, service application **106** queries each connected data source to determine access syntax and format. Service application **106** thus generates and stores metadata for retrieving data stored in the connected data sources.

After startup processing, desktop client **105** is copied from (downloaded from) server computer **102** to client computer **101** for execution. The copying and execution of desktop client **105** is initiated from within browser **104**. Upon execution, desktop client **105** presents the user with a desktop window. Figure 4 illustrates a desktop window **400** in accordance with the present invention. The desktop window **400** presents a user with the option to select a set of books **402** that are available to the user. A book is representative of an information link to one of the data sources in data source set **108** and the applications used to process such data. Because desktop window **400** is provided by desktop client **105** running within browser **104**, the data source set **108** is advantageously accessed without dependence on the hardware platform and operating system included in client computer **101**.

Desktop window **400** provides access to the data associated with each book **402** (e.g., enables the viewing of data associated with the book). Access is controlled by disabling selected items in a dropdown menu **404** that appears when a user selects a book **402**. Selecting the open command from dropdown menu **404** causes a view manager dialog window **500** as shown in Figure 5 to be presented. View manager dialog window **500** is a dialog box that displays a

list of available views **501** for a corresponding book **402**. Selecting a particular view **501** causes data associated with the corresponding book **402** to be displayed in accordance with the format associated with the selected view **501**. Desktop window **400** also includes a status bar that presents operational and status messages along the bottom of desktop window **400**.

Books are added and deleted by user or system administrators using interfaces presented from desktop client **105**. To add (create) a new book, a user selects a "New" command under "Books". This selection causes a new book properties window, **600** as shown in Figure 6A to appear. The new book properties window **600** includes a services box **601** that enables a user to select the service (e.g., the server) that contains the data source that the user wants to associate with the book. After a service is selected, the desktop client **105** next presents an updated new properties window **602** as shown in Figure 6B. The updated new properties window **602** replaces services box **601** with a datasource box **603**. Data source box **603** includes the name of dates sources (e.g., JDBC, MAP1, POP3, Native etc.) that are available from the selected service.

After selecting one of the presented data sources, desktop client **105** presents a set of dialog boxes that enable a user to further specify information for accessing the corresponding data source. For example, if a JDBC data source is selected, the user is presented with a dialog box such as the further updated new book properties window **605** shown in Figure 6C. Further updated window **605** includes a new box **606** titled "driver". The further updated window **605** provides a list of JDBC drives corresponding to configured JDBC drivers for selection. After selecting a driver, desktop client **105** presents either dialog boxes enabling a uses to select a database by name and a table within the selected database.

If a user selects an electronic mail data source rather than a JDBC source from updated new properties window **602**, desktop client **105** presents a user with a sequence of dialog boxes allowing the user to select the desired mail profile and its location.

Alternatively, books are added to a desktop window **200** by importing them from a server **102**. To import a book, a users uses a set of dialog boxes to select a server computer **102** and to the select a book on that server.

The following user's manual is illustrative of and further describes system and method of the present invention:

Sanga Pages: Version 1.0

Pages Desktop Client - As an implementation of browser/server computing, the Pages Desktop client is an applet that gives Java-enabled browsers a window into Sanga Pages applications.

Pages Desktop Server - The core of Sanga Pages, it allows clients to access "books of information" which represent links to data sources and the applications required to leverage that data.

- Access controlled by user ID/password
- "Book" metaphor for retrieving, storing and viewing Sanga Forms and Views
- Network tools for remote access to personal customized Desktop Client through Internet and access to databases on other servers
- Administrative tools for adding users and groups, broadcasting messages, and more

Pages Form Design - Gives developers a set of GUI design components to create sophisticated front-ends for custom applications or templates for data viewing. Standard GUI controls include static text boxes, buttons, text fields, scrolling text, check boxes, radial buttons, pull down lists, and regular lists. All controls have selectable color, fonts and scripts. Also includes variable granularity grids for alignment, group alignment, and resizing. Includes standard reusable scripts.

Pages Script - This robust, BASIC-like compiled programming language enables developers to add functionality to forms and create more detailed custom applications. Together with Pages View Design and Pages Form Design this completes the package of platform-independent Java application development tools. Scripting options include declaring variables of different types, executing common looping and conditional expressions, creating extensions into the Sanga

Pages API library, displaying message boxes, performing field access and validation, inserting, updating, deleting data, database navigation (next, last, previous etc.) syntax checking and error reporting.

Pages Security - Administrators can create access control lists and use DES encryption of sessions. Developers can also use wrappers that support Northern Telecom's Entrust for industrial-strength security. Sanga Pages Security can encrypt all data transmitted to and from Sanga Pages clients and SQL databases, provide user-level security, field level security, and support digital signatures.

Pages Mail - This supports and integrates all major protocols (including POP3/ SMTP, MAPI and IMAP4) with attachment support.

Pages Replication - Sanga Pages allows users to copy databases across platforms for efficient distributed-database applications and to update different copies of the same database.

Pages View Design - Sanga provides a flexible query tool that enables users to create custom ad hoc views of data. This tool is perfect for managing large amounts of data and letting end-users filter and present information from SQL databases right from within their browser. Operands supported include <,>,<>,= and , or, like. Users can specify which database fields to view and adjust the order, name, and column width of those fields.

Contents

Introduction

This manual introduces you to Sanga Pages Version 1.0. It contains all the information and instructions you need to use Pages Desktop Client, Pages Server, Pages View Design, Pages Form Design, Pages Script, and Pages Security.

This manual assumes you are familiar with either Netscape Navigator 2.0 (or better), Microsoft Internet Explorer 3.0 (or better), or other equivalent web browsers capable of running Java applets.

Chapter Descriptions

The following list outlines the chapters in this manual and their contents:

"Introduction", which you are currently reading, introduces the manual and the features of Sanga Pages.

"Starting Pages Desktop Client" describes how to start and exit Pages Desktop, and reviews the components of the Pages Desktop window.

"Quick Start" provides instructions on how to accomplish basic tasks in Pages Desktop.

"Pages Desktop Commands" describes in detail the commands available in Pages Desktop.

"Pages View Design" describes in detail how to create and edit a view of a Pages Book.

"Pages Form Design" describes in detail how to create and edit a form used to examine the fields of a Pages Book.

"Pages Script" describes in detail the syntax of the Pages Script language.

"Pages Security" explains the security features available in Sanga Pages.

"Pages Server" describes in detail the features and tools available to an administrator of Sanga Pages.

Manual Conventions

This manual uses certain conventions, including the typographical conventions in the manual itself, keyboard conventions for using Sanga Pages and the terminology describing mouse techniques.

Common Actions

This manual uses the following terms to mean specific actions:

Select

Select means you indicate an item by clicking once on the item to move the highlight to the item. For example:

- you select one option button from a group of option buttons;
- you select an item in a drop-down box;
- you select a file name in a dialog box; or
- you select a check box.

Choose

Choose means you carry out an action. This means double-clicking on an item, or pressing ENTER once the item is selected. Choosing an item causes something to happen immediately. For example:

- you choose a command from a menu;
- you choose a command button in a dialog box;
- you choose OK in a dialog box to activate any changes you have made; or
- you choose CANCEL to close a dialog box without making any changes.

Typographical Conventions

This manual uses various typefaces, including:

SMALL CAPS
Entries in small caps represent specific keys on the keyboard that you press. Graphic symbols or abbreviations may label these keys.

`Monospaced`
Monospaced type indicates a command that you type.

Italic
Italic type means you replace the text in italics with your own text.

[square brackets]
Items in square brackets are options in commands

Bold
Menu selections, commands, and controls in dialog boxes use bold type.

Mouse Techniques

A one or two word term identifies commonly used actions. In this manual, these terms are used to avoid repeating detailed descriptions of the action. These

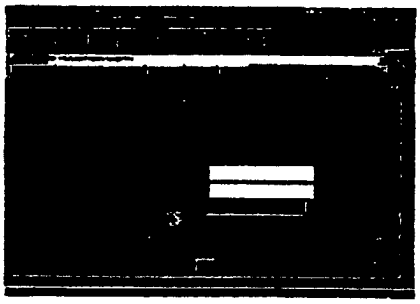
terms are:

- point means you move the mouse pointer on the screen until it rests on the command or item you want to select.
- click refers to the action of pressing and releasing the mouse button quickly. The instruction to click something means you use the left button on the mouse, while right-click means you must use the right button.
- double-click means you press the left mouse button twice while on the indicated object.
- drag means you press and hold the mouse button while moving the mouse.

Starting Pages Desktop Client

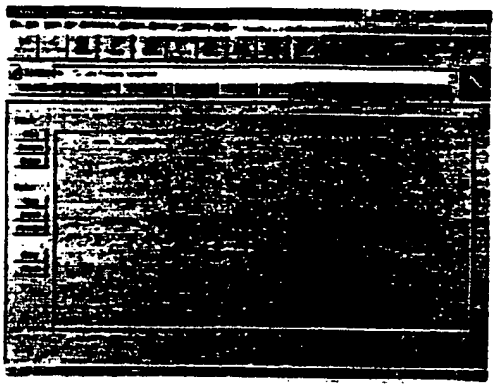
This chapter introduces you to Pages Desktop Client. It describes how to start and exit the application, and reviews the components of the Pages Desktop window. For the purposes of this manual, Pages Desktop Client will be referred to as Pages Desktop.

To start Pages Desktop, open your preferred Java-enabled web browser (i.e. Netscape Navigator, Microsoft Internet Explorer, et cetera) and call up the Pages Desktop site. The Pages Logon window will appear asking you for your user id, password, and desktop service. Enter both your *user id* and *password* and select the desktop service you wish to use from the dropdown list (the default value is to use the local service). Select OK to continue. If you do not wish to log on at this time then either close your web browser or open a new location to end the Logon session.



If you select OK and the *user id* and *password* are valid, the Pages Desktop window will appear in the default setting developed by your Sanga Pages administrator.

Pages Desktop Window



Pages Desktop uses a tabular layout common to many Windows applications. This discussion assumes a standard installation with a default configuration.

The main portion of the Pages Desktop is the tabbed panel used to organize the books of a Pages user. Initially there will be one tab without a title.

The Button Bar appearing down the left side of the window shows the initial selections available. Under "Tabs", New, Rename, and Delete appear. Under "Books", the buttons called New, Import and Arrange appear. Save and About are the last two buttons.

The Status Bar, which gives you operational and status messages, appears along the bottom of the Pages Desktop window.

Quick Start

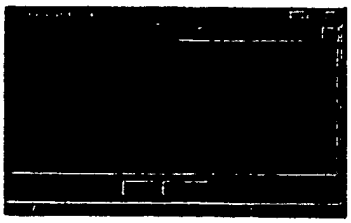
This chapter gets you started using Pages Desktop for some basic tasks, including viewing data in forms and entering data into forms. These instructions provide you with the ability to get up and running with Pages Desktop quickly.

All of the instructions in Quick Start assume that your administrator has already set up your desktop, provided you with Pages Books you can import and you are able to Log onto to your Pages Desktop. If you are setting up your own desktop, please refer to the chapter entitled "Pages Desktop Commands".

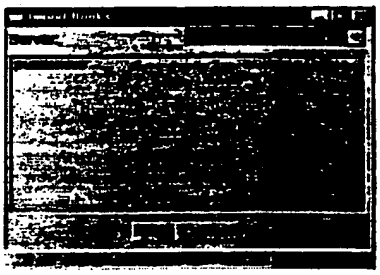
Importing a Pages Book

This will show you how to quickly import a Pages Book to get working right away.

1. On the main Pages Desktop select the Import button under the "Books" heading. This will bring up the import books dialog window.



2. From the dropdown list of servers (Pages Servers) choose the same one you logged onto (unless your administrator has placed the default books in another location). A list of the Pages Books available for import appears in the selection list.

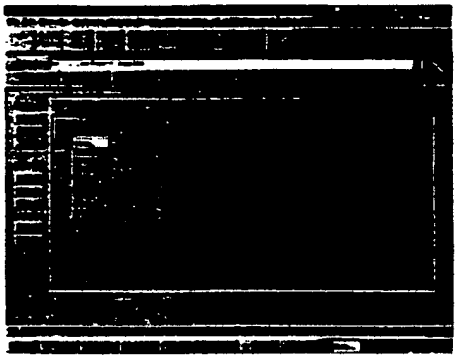


3. Select a book you wish to import from the list and click on the OK button. The dialog window should disappear and the selected tab in the Pages Desktop should now contain the Pages Book you just imported.

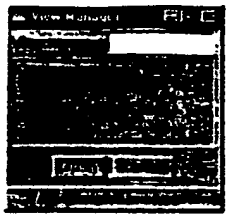
View the data in the Pages Book

With the Pages Book you just imported, you should be able to use the Views and Forms to inspect the data. To view the data in a Pages View:

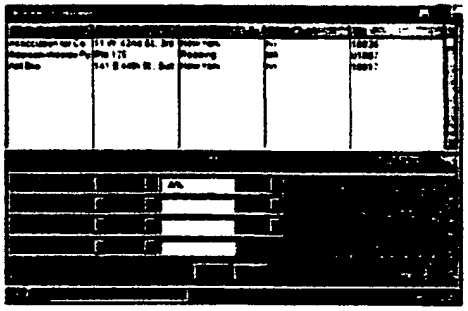
1. Click on the Pages Book to bring up its dropdown menu.



2. Select the Open... item. This will bring up the View Manager dialog window which will display a list of available Pages Views.



3. Select the View you wish to use from the list and click the Open button. This will bring up a View window containing data from the Pages Book.

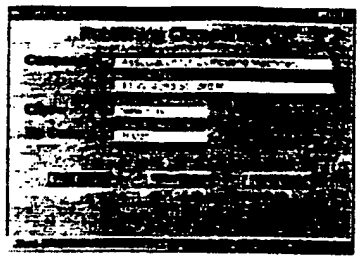


To view the data in a Pages Form:

1. Click on the Pages Book to bring up its dropdown menu.

2. Select the New... item. This will bring up the Form Manager dialog window which will display a list of available Pages Forms.

3. Select the Form you wish to use from the list and click the Open button. This will bring up a Form window containing data from an entry in the Pages Book.



Field Help

When you place your cursor on a button in Pages Desktop, the help field under the workspace area will indicate the function of the button. For example; if you place your cursor on the New button underneath "Tabs", you will see the message "Create new workspace" beneath the current workspace.

Creating a New Tab

1. In the Pages Desktop window, click on the New button under "Tabs". A new window will appear called "New Tab".

2. In the text box following Desktop Name, enter the *name* you want to call the tab.

3. Select the OK button to save the chosen tab name or Cancel to return to the Pages Desktop window.

4. If you select OK, a new tab will appear in the Pages Desktop window with the name you have chosen.

Renaming a Tab

1. In the Pages Desktop window, select the tab you wish to rename by clicking on the tab name. This action will move the tab to the front of the other tabs.

2. Click on the Rename button under Tab. A new window will appear called "Rename Tab".

3. In the text box following New Desktop Name, enter the new tab name.

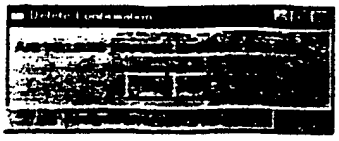
4. Select the OK button to save the new tab name or Cancel to return to the Pages Desktop window.

5. If you select OK, the original tab will appear with the new name you have chosen.

Deleting a Tab

1. In the Pages Desktop window, select the tab you wish to delete by clicking on the tab name. This action will move the tab to the front of the other tabs.

2. Click on the Delete button under Tabs. A new window will appear called "Delete Confirmation". The message in the window is "Are you sure you want to delete this tab?"



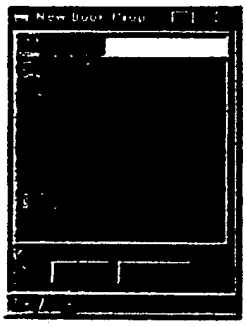
3. If you select the No button, the request for deletion will be canceled and you will return to the Pages Desktop window.

4. If you select the Yes button, the tab selected will be deleted, including the books within the tab.

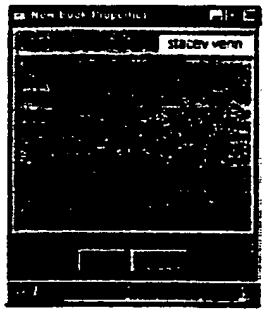
Creating a New Book

1. In the Pages Desktop window, select New under "Books".

2. A new window will appear entitled "New Book Properties". In the "Services" box, select the service that contains the data source you want this Pages Book to reference. Either double-click on the selected service name or select Open from the bottom of the window.



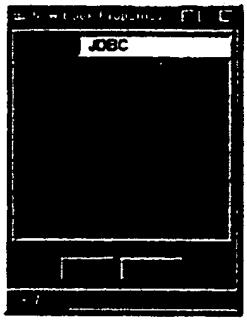
3. A new box will replace the "Services" box called "Pages Data Source". In the box are all the names of the data sources (i.e.: JDBC, MAPI, POP3, Native) available to you from the service you just selected. Select the data source you wish to use for your Book. Then select **Open**.



4. Depending upon the type of data source you selected, the remaining dialogs will differ as follows:

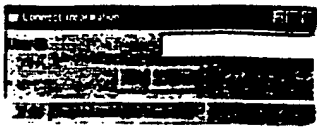
   JDBC

   a) Another new box will replace "Pages Data Source" called "Driver". Select the JDBC driver (these will correspond to the configured ODBC drivers) you wish to use for your Book and then select **Open**.

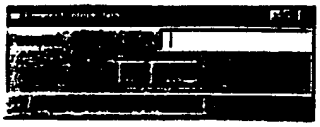


14 SUBSTITUTE SHEET

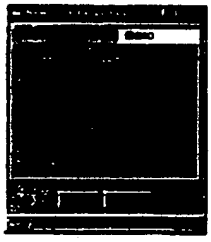
b) A new window appears called "Connect Information". In the text box following "User ID", enter your user id and then select OK.



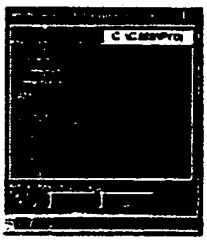
c) A new window appears requesting your password. Enter your password in the text box and then select OK.



d) The next message in the "New Book Properties" window is "Database Name." Select your chosen database and then click on Open.

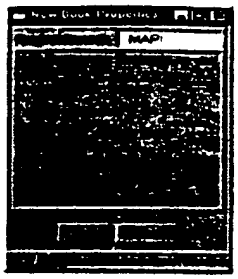


e) A new message appears in the New Book Properties" window asking for the "Table Name". Select the table name you wish this book to represent and then select Open.



MAPI

a) Another new box will replace "Pages Data Source" called "Profile Name". Select the mail profile you wish to use for your Book and then select Open.



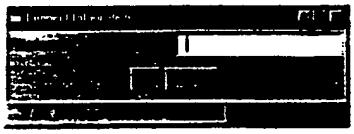
POP3

a) A new window appears called "Connect Information". In the text box following "Mail (POP3) Server", enter the address of the server and then select OK.

b) A new window appears requesting the location of your SMTP Server. Enter this information in the text box and then select OK.

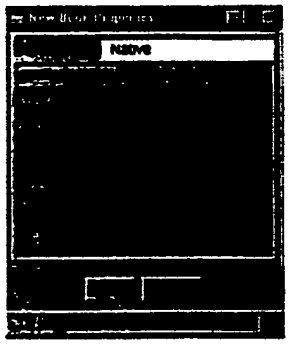
c) A new window appears requesting your user name. In the text box following "User Name", enter your user id for the mail server and then select OK.

d) A new window appears requesting your password. Enter your password in the text box and then select OK.

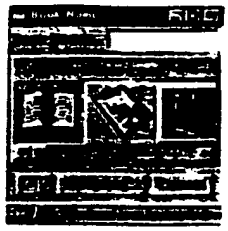


Native

a) Another new box will replace "Pages Data Source" called "DataBase". Select the native database you wish to use for your Book and then select Open.



5. A new window appears called "Book Name". Enter a name for your book, select an icon to represent the book and then click OK.



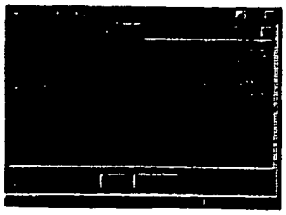
6. The new Pages Book icon labeled with the name you have chosen will appear in the current workspace in Pages Desktop.

Importing a Book

This feature allows you to import an existing Pages Book into your Desktop. The book you wish to import can exist either on your local service or any other available service in the network.

1. Select the Import button under "Books.

2. A new window appears called "Import Books". Select a server (Pages Server) from the dropdown box following the prompt.(insert import1.bmp)



3. A list of all the Pages Books available in that server will appear in the box underneath the Server prompt.

4. Select the book you wish to import to your desktop. Click OK to import the book and return to your main desktop window. The imported book icon will appear in your current workspace.

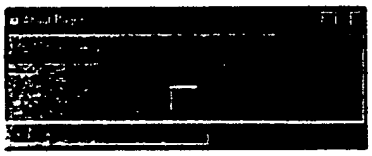
5. Select Cancel to return to the main desktop window without importing the book.

Save

If you select the Save button, you will save the format of your desktop as well as all the books and forms you have created. It is suggested that you save your desktop periodically during the time you use it to prevent losing the forms and views.

About

If you select the About button information about product, including the version number will be displayed in a window. Click on the OK button to dismiss the window.



Working With a Book

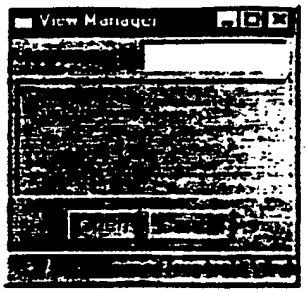
To work with a Pages Book, click once on the book's icon. A drop-down menu will appear with the following choices:

Open

This option allows you to open a previously created Pages View of the book.

1. Select Open... from the Book's dropdown menu.

2. A new window appears called "View Manager". This will present a list of available Pages Views for the book.



3. To select a view either type the name in the text box following the "View Name" prompt or select a view from the list.

4. Click on the Open button to continue or Cancel to return to the main Pages Desktop window.

5. If you selected Open a new window appears entitled "View - *ViewName*", where *ViewName* matches the one selected in step 3. You cannot edit this view but you can use the Query tool on the data. The Query tool allows you to filter the data of the view by applying a set of selection criteria. With the tool you can filter on a single database field or a combination of up to four. There are three elements to each selection criteria:

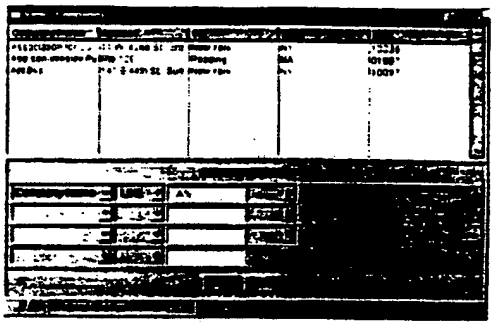
   a) Database Column - this is a list of all columns in the database

   b) Comparison - the tools for comparison are equals (=), not equals (<>), less than (<), greater than (>) and like (LIKE).

   c) Value - this is the value to compare against.

   For example one selection criterion applied to an *Employee* database may be: `YearsOfService > 5`. This would return a view of all employee records in the database with a value in the YearsOfService column greater than five.

The selection criteria can be joined with either AND or OR. If AND joins two selection criteria then both must return true for the record to be displayed in the view. Using OR in the filter list means that a record which matches any of the selection criteria will be included in the view.



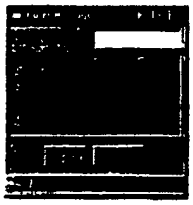
6. Close the window to return to the main Pages Desktop window.

New

This option allows you to open a previously created Pages Form for viewing the entries in a Pages Book.

1. Select New... from the Book's dropdown menu.

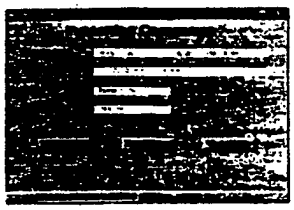
2. A new window appears called "Form Manager". This will present a list of available Pages Forms for the book.



3. To select a form either type the name in the text box following the "Form Name" prompt or select a form from the list.

4. Click on the Open button to continue or Cancel to return to the main Pages Desktop window.

5. If you selected Open a new window appears entitled "Pages Form". You cannot edit the layout of this form but a properly created form should allow you interaction with the data.



Design:Views...

This option allows you to create, edit, and delete views of the Pages Book.

1. Select Design and then Views... from the dropdown menu. This will bring up the "View Manager" window used for working with Pages Views.

For further information on accessing the functionality of the "View Manager" window, please refer to the chapter entitled "Pages View Design".

Design:Forms...

This option allows you to create, edit, and delete forms of the Pages Book elements.

1. Select Design and then Forms... from the dropdown menu. This will bring up the "Form Manager" window used for working with Pages Forms.

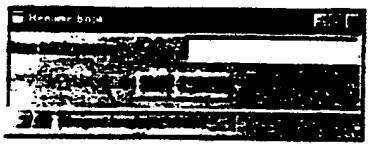
For further information on accessing the functionality of the Pages Form Design, please see that chapter.

Rename

This option allows you to rename the Book.

1. Select Rename from the drop-down menu.

2. A new window entitled "Rename Book" appears. In the text box following the message New book name, type in the new name for this book.



3. If you are satisfied with your new name, click OK to return to the main menu. You will see that the new name of the book has replaced the old name under the icon.

4. If you decide not to rename the book, click the Cancel button to return to the main drop-down menu.

Copy

This option is not part of Sanga Pages Version 1.0

Replicate

This option is not part of Sanga Pages Version 1.0

Delete

This option allows you to delete the selected Pages Book.

1. Select Delete from the dropdown menu.

2. A confirmation screen will appear asking you "Are you sure you want to remove the selected book?"

3. Select Yes to confirm the deletion and the Pages Book will be removed from the Desktop.

4. Select No to cancel the deletion of the book.

Icon:Stick and Icon:UnStick

These options allow you to share and/or move Pages Books between the tabs of the Desktop.

1. To share a Pages Book among the tabs of your Desktop, simply select Icon and then Stick from the book's dropdown menu. Now if you move among the various tabs of your Desktop you will see book is visible to all of them.

2. If you want to move a Pages Book from one tab to another begin by following the same procedure in step 1. Now move to the tab where you wish to move the book to and select Icon and then UnStick from the book's dropdown menu. Now the book is only visible onthat tab.

Properties:Show

This option shows you all the Book's properties (i.e. data source, driver, user name, et cetera)

1. Select Properties and then Show from the book's dropdown menu.

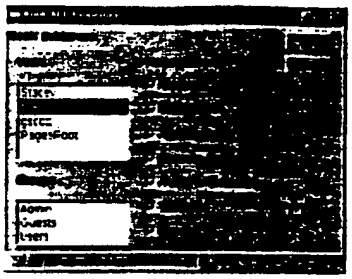
2. A new window appears called "Book Properties". This window contains information about the book you have opened, including the name and the Pages Data Source. Other information displayed depends on the type of Pages Data Source the book is linked to.

3. Select OK or Cancel to return to the main Pages Desktop window.

Properties:ACLs...

This option allows you to create an Access Control List (ACL) for your Pages Book. Creating an ACL will ensure that only authorized users can import your book.

1. Select Properties and then ACLs... from the book's dropdown menu. A new window appears called "Book ACL Properties".



2. The Name of the book is displayed at the top. This value cannot be changed as it must match the Pages Book name.

3. Below this are two lists: a User List and a Group List. The User List contains all local users managed by this service. The Group List contains the default system wide groups as well as any locally defined groups.

4. Next to these lists are the Pages Permissions which are made up of the following set (Control, Read, Update, Execute, Test for Existence, Delete, Create). Whenever a user or group is selected these check boxes will reflect the permission set for that user/group. To add (box contains a ✓) or remove (box is empty) a permission simply click on the appropriate field and the value will be toggled.

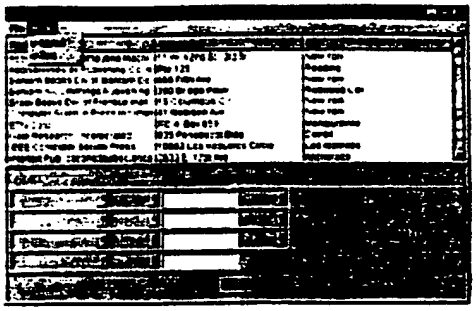
5. Select OK or Cancel to close this window and return to the Pages Desktop.

Pages View Design

Creating a View

1. Select Design and then Views... from the Book's dropdown menu.
2. A new window appears called "View Manager".
3. Enter the name of the view you wish to create in the text box following the prompt for View Name.
4. Click on the Create button. Two new windows, similar to those shown below [viewdesign1.bmp and viewtools1.bmp], appear entitled "View Tools" and "View Design - *viewName*", where *viewName* is the name you entered in step 3.
5. In the "View Tools" window, select a field that you want included in the view from the Column list.
6. The Name of the field will appear in the text box following Label: in the Properties box. You can change the label of the column in the view by deleting the default name for label and then entering one of your choice.
7. To display the selected field in the view you are creating, select the Display in view check box. You will notice that there is a new column in the "View Design" window whose title matches the label of step 6.
8. Continue to select fields for the view you are designing by following steps 5 to 7. Each field that is added when the Display in view check box selected will appear in the "View Design" window. Once you have selected all the fields needed for your view, you may now test the view design.

9. When adding fields to the view, the Mode menu on the "View Design" window should have Layout selected. To test accessing and displaying information in your view, select the Test option under the Mode menu on the "View Design" window. Testing the view will display the information of the columns you have selected from the book.



10. To save the current view design at any time, select Save under the File menu to save the design under it's current name. Select Save As... under the File menu to bring up the "Save As..." dialog window which will allow you to save the design with a different name or overwrite an existing view design.

11. To exit the "View Design" window at any time, select Close Editing from the File menu on the "View Design" window.

Editing a View

1. Select Design and then Views... from the Book's dropdown menu.

2. A new window appears called "View Manager".

3. Select the view that you wish to edit. The view name will appear in the text box following the View Name prompt.

4. Click on the Edit button. Two new windows appear entitled "View Tools" and "View Design - *viewName*", where *viewName* is the name you selected in step 3.

5. In the "View Tools" window, select the field that you wish to edit in the view from the Column list.

6. The name of the field will appear in the text box following Label in the Properties box. You can edit the label of the column by deleting the current label name and then entering a new name for the column.

7. To change the field display of the view, select or de-select the Display in View check box, depending on the selected field. You will notice the field and label changes happening concurrently in the "View Design" window.

8. Continue to edit the view you are designing by following steps 5 to 7. Once you have finished editing the columns in your view, return to the "View Design" window.

9. When editing fields in the view, the Mode menu on the "View Design" window should have Layout selected. To test accessing and displaying information in your view, select the Test option under the Mode menu on the "View Design" window.

10. To save the current view design at any time, select Save under the File menu to save the design under it's current name. Select Save As... under the File menu to bring up the "Save As..." dialog window which will allow you to save the design with a different name or overwrite an existing view design.

11. To exit the "View Design" window at any time, select Close Editing from the File menu on the "View Design" window.

Deleting a View

1. Select Design and then Views... from the Book's dropdown menu.

2. A new window appears called "View Manager".

3. Select the view that you wish to delete. The view name will appear in the text box following the View Name: prompt.

4. Click on the Delete button. A new window will appear called "Delete Confirmation". The message in the window is "Are you sure you want to delete the selected item?"

5. If you select either the No or Cancel button, the request for deletion will be canceled.

6. If you select the Yes button, the selected view will be deleted.

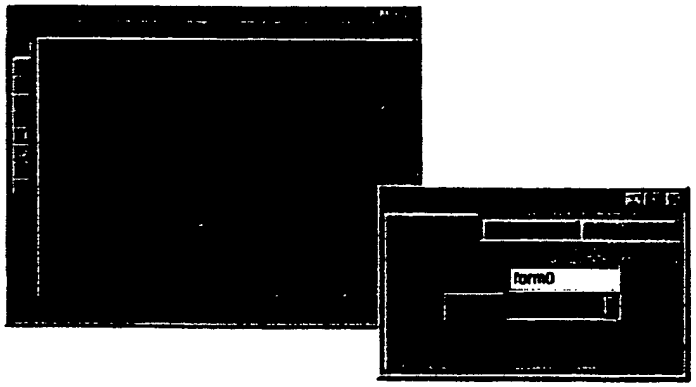
2. ages View ...

7. After confirming the deletion you will return to the "View Manager" window where you can select Cancel to close the window.

Pages Form Design

Creating Forms

1. Select **Design** and then **Forms...** from the Book icon's dropdown menu.
2. A new window appears called "Form Manager".
3. Enter the name of the form you wish to create in the text box following the prompt for **Form Name**.
4. Click on the **Create** button. Two new windows, similar to those shown below, appear entitled "Properties" and "Form Design".



5. In the "Form Design" window you can create several different types of objects in your form:

Creating a Text Label in Form Design

1. In the "Form Design" window, click on the Create text label button, or select Label from the tools menu.

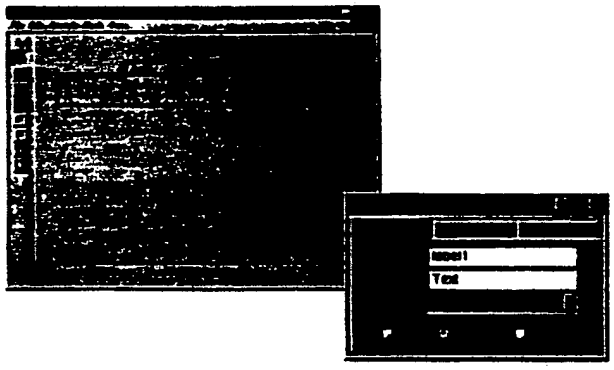
2. In the "Form Design" window again, click and drag to create an appropriate-sized text. The text box can be resized at any time by selecting a resize tab on the text box and dragging it to the desired size.

3. By default the word *Text* will appear in the label box you have created.

4. Select the General tab of the "Properties" window. The default values of the text label will appear; the Id for the text label will be "label#", Label will contain the actual text for the label, with the default of *Text*, and Field will be "[none]".

5. To change the Id of the text label, simply enter a different ID instead of the default value following the "Font" label in the "Properties" window.

6. To change the text inside the label, simply replace *Text* in the text box following the prompt for Label in the "Properties" window.



7. To adjust the font size and style of the text in the text label, click on the Font tab in the "Properties" window.

8. In the Font tab, select your chosen font from the dropdown list following the Font: prompt.

9. Enter your selected font size in the text box following Font Size. You also have the option of making your font in the text label box bold and/or italic by selecting the appropriate check boxes.

10. To adjust the color of the text and the background of the text box, select the Color tab in the "Properties" window.

11. To change the color of the text, select the Foreground radio button. Then point and click your cursor on the selected color in the color box. You will notice that the text in the "Form Design" window's text label will change to the selected color.

12. To change the color of the background of the text label, select the Back ground radio button in the Color tab. Then point and click your cursor on your chosen color in the color box. You will notice that the background of the text label in the "Form Design" window will change to the selected color.

Creating a Button

1. In the Form Design window, click on either the Create button tool button, or Button from the Tools dropdown menu.

2. Click and drag to create an appropriate-sized button in the "Form Design" window. A three-dimensional button will appear with the default label of *Button*.

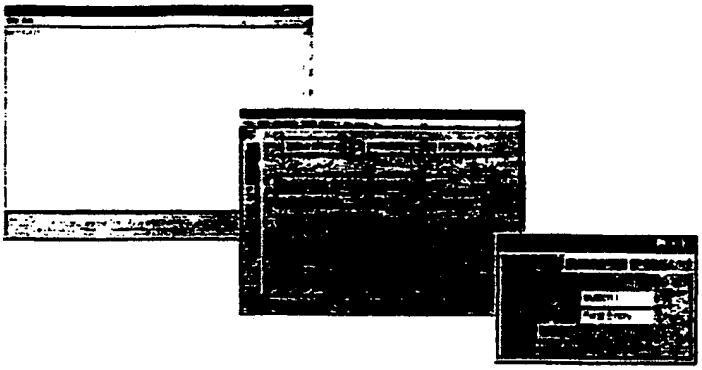
3. In the "Properties" window, select the General tab. The default values of the button will appear; Id will be "button#", Label will be *Button*, and the Script button will reveal itself to be empty.

4. To change the Id of the Button, simply enter in a different ID of the button rather than the default value.

5. To change the label of the button from the default setting of *Button*, simply enter the new button label in the text box following the prompt for Label.

6. To add the script which will enable the button to perform its desired action, either select a database utility routine from the dropdown list or click on the Script button.

7. A new window appears called "Script - null". Enter the script code which enables the button to perform its desired action in the form. Remember to enter a carriage return after each line. Please refer to the chapter on Pages Script for details on programming in Script.

8. 

9. To adjust the font type and style of the text in the Button, click on the Font tab in the "Properties" window.

10. You can change the font of the button using the dropdown box to choose a font.

11. The font size is automatically set when you create the button. You have the options of making the button label bold and/or italic by selecting the appropriate check box.

Note:

Unlike text boxes, you cannot change the color of the text of the button label or the button itself.

Creating Single-Line Text

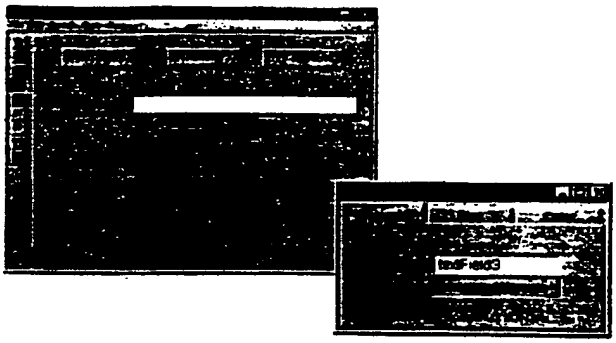
1. In the "Form Design" window, click on either the Create single-line text toolbutton or Text Field from the Tools dropdown menu.

2. In the "Form Design" window again, click and drag to create an appropriate-sized text box.

3. Select the General tab of the "Properties" window. The default values of the text box will appear; Id will be "textField#" and Field will be "[none]".

4. To change the Id of the text box, simply enter a different ID instead of the default value.

5. Select the field of the book to be displayed in the text box by choosing one from the Field dropdown list.

6. 

7. To adjust the font size and style of the text in the text box, click on the Font tab in the "Properties" window.

8. In the Font tab, select your chosen font from the dropdown list following the Font prompt.

9. Enter your selected font size in the text box following Font Size. You also have the options of making your font in the text box bold and/or italic by selecting the appropriate check boxes.

10. To adjust the color of the text and the background of the text box, select the Color tab in the "Properties" window.

11. To change the color of the text, select the Foreground radio button. Then point and click your cursor on the selected color in the color box.

12. To change the color of the background of the text label, select the Back ground radio button in the Color tab. Then point and click your cursor on your chosen color in the color box. You will notice that the background of the text box in the "Form Design" window will change to the selected color.

Creating a Scrollable Text Area

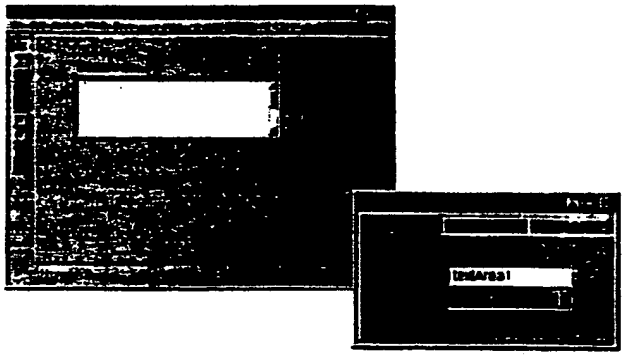
1. In the "Form Design" window, click on either the **Create scrollable text area** tool button or **Text Area** from the **Tools** dropdown menu.

2. In the "Form Design" window again, click and drag to create an appropriate-sized scrollable text area. In general these areas will be taller then the single-line text areas so as to display several lines of text at once.

3. Select the **General** tab of the "Properties" window. The default values of the text area will appear; **Id** will be "textArea#" and **Field** will be "[none]".

4. To change the Id of the text area, simply enter a different ID instead of the default value.

5. To select a field from the book for the text area, choose one from the dropdown list following **Field**.

6. 

7. To adjust the font size and style of the text in the text box, click on the **Font** tab in the "Properties" window.

8. Select your chosen font from the dropdown list following the **Font** prompt.

9. Enter your selected font size in the text box following **Font Size**. You also have the option of making your font in the text box bold and/or italic by selecting the appropriate check boxes.

10. To adjust the color of the text and the background of the text box, select the **Color** tab in the "Properties" window.

11. To change the color of the text, select the **Foreground** radio button. Then point and click your cursor on the selected color in the color box.

12. To change the color of the background of the scrollable text area, select the **Background** radio button in the Color tab. Then point and click your cursor on your chosen color in the color box. You will notice that the background of the scrollable text area in the "Form Design" window will change to the selected color.

Creating a Checkbox

1. In the "Form Design" window, click on either the **Create checkbox** tool button or **Checkbox** from the **Tools** dropdown menu.

2. In the "Form Design" window again, click and drag to create an appropriate-sized checkbox.

3. Select the **General** tab of the "Properties" window. The default values of the text box will appear: **Id** will be "checkBox#", the **Label** will be *Checkbox*, and **Script** will be empty.

4. To change the Id of the checkbox, simply enter a different ID instead of the default value.

5. To change the **Label** of the checkbox, replace the original *Checkbox* label and enter one of your choice.

6. To add the script which will enable the checkbox to perform its desired action, either select a database utility routine from the dropdown list or click on the **Script** button.

7. A new window appears called "Script - null". Enter the script code which enables the checkbox to perform its desired action in the form. Remember to enter a carriage return after each line. Please refer to the chapter on Pages Script for details on programming in Script.

8. To adjust the font size and style of the label of the checkbox, click on the **Font** tab in the "Properties" window.

9. Select your chosen font from the dropdown list following the **Font** prompt.

10. Enter your selected font size in the text box following Font Size. You also have the options of making the font of the checkbox label bold and/or italic by selecting the appropriate check boxes.

11. You cannot adjust the color of the checkbox.

Creating a Radio Button

1. In the "Form Design" window, click on either the Create radio button tool button or Radio Button from the Tools dropdown menu.

2. In the "Form Design" window again, click and drag to create an appropriate-sized radio button including a text label.

3. Select the General tab of the "Properties" window. The default values of the radio button will appear; Id will be "radioButton#", the Label will be *Radio Button*, the Group will be empty, and Script will be empty.

4. To change the Id of the radio button, simply enter a different ID instead of the default value.

5. To change the Label of the radio button, replace the original *Radio Button* label and enter one of your choice.

6. For any buttons in the same group, at most one of the buttons will be active at one time. When a button is selected, the previously selected button is de-selected. If it is required for several radio buttons to provide toggles where at most one of the buttons can be active at one time, radio buttons should be added to groups. In the Group section you must define the group this radio button belongs to.

7. To add the script which will enable the radio button to perform its desired action, either select a database utility routine from the dropdown list or click on the Script button.

8. A new window appears called "Script - null". Enter the script code which enables the radio button to perform its desired action in the form. Remember to enter a carriage return after each line. Please refer to the chapter on Pages Script for details on programming in Script.

9. To adjust the font size and style of the text in the label text box, click on the Font tab in the "Properties" window.

10. Select your chosen font from the dropdown list following the Font prompt.

11. Enter your selected font size in the text box following Font Size. You also have the options of making your font in the text box bold and/or italic by selecting the appropriate check boxes.

Note:

You cannot adjust the color of a radio button.

Creating a Dropdown List

1. In the "Form Design" window, click on either the Create dropdown list tool button or Choice from the Tools dropdown menu.

2. In the "Form Design" window again, click and drag to create an appropriate-sized box for a dropdown list.

3. Select the General tab of the "Properties" window. The default values of the dropdown list will appear; Id will be "choice#" and the Script will be empty.

4. To change the Id of the dropdown list, simply enter a different ID instead of the default value.

5. To add the script which will enable the dropdown list to perform its desired action, click on the Script button.

6. A new window appears called "Script - null". Enter the script code which enables the dropdown list to perform its desired action in the form. Remember to enter a carriage return after each line. Please refer to the chapter on Pages Script for details on programming in Script.

7. To adjust the font size and style of the text in the dropdown list, click on the Font tab in the "Properties" window.

8. Select your chosen font from the dropdown list following the Font prompt.

9. Enter your selected font size in the text box following Font Size. You also have the options of making your font in the text box bold and/or italic by selecting the appropriate check boxes.

10. To adjust the color of the text and the background of the text box, select the Color tab in the "Properties" window.

11. To change the color of the text, select the **Foreground** radio button. Then point and click your cursor on the selected color in the color box.

12. To change the color of the background of the text label, select the **Back ground** radio button in the Color tab. Then point and click your cursor on your chosen color in the color box. You will notice that the background of the dropdown list in the "Form Design" window will change to the selected color.

Creating a Scrollable List

1. In the "Form Design" window, click on either the **Create scrollable list** tool button or **List** from the **Tools** dropdown menu.

2. In the Form Design window again, click and drag to create an appropriate-sized scrollable list box.

3. Select the **General** tab of the "Properties" window. The default values of the scrollable list will appear; Id will be "form#" and Script will be empty.

4. To change the Id of the scrollable list, simply enter a different ID instead of the default value.

5. To add the script which will enable the scrollable list to perform its desired action, click on the **Script** button.

6. A new window appears called "Script - null". Enter the script code which enables the list to perform its desired action in the form. Remember to enter a carriage return after each line. Please refer to the chapter on Pages Script for details on programming in Script.

7. To adjust the font size and style of the text displayed in the scrollable list box, click on the **Font** tab in the "Properties" window.

8. Select your chosen font for the text in the scrollable list from the dropdown list following the **Font** prompt.

9. Enter your selected font size in the text box following **Font Size**. You also have the options of making your font in the text box **bold** and/or *italic* by selecting the appropriate check boxes.

10. To adjust the color of the text and the background of the scrollable list box, select the **Color** tab in the "Properties" window.

11. To change the color of the text select the Foreground radio button. Then point and click your cursor on the selected color in the color box.

12. To change the color of the background of the scrollable list box, select the Background radio button in the Color tab. Then point and click your cursor on your chosen color in the color box. You will notice that the background of the scrollable list box in the "Form Design" window will change to the selected color.

Editing Forms

1. Select Design and then Forms... from the Book icon's dropdown menu.

2. A new window will appear called "Form Manager". A list of possible Pages Forms to select from are in the list under the prompt for Form Name.

3. Click on the form you wish to edit. The name of the form you have chosen will appear in the text box following Form Names.

4. Select Edit if you wish to use Pages Form Design to edit your selected form.

5. Two new windows will appear called "Properties" and "Form Design". Edit the Form using the same directions as the section on "Creating a New Form".

Deleting a Form

1. Select Design and then Forms... from the Book's dropdown menu.

2. A new window appears called "Form Manager".

3. Select the form that you wish to delete. The view name will appear in the text box following the Form Name prompt.

4. Click on the Delete button. A new window will appear called "Delete Confirmation". The message in the window is "Are you sure you want to delete the selected item?"

5. If you select either the No or Cancel button, the request for deletion will be canceled.

Pages Form

6. If you select the Yes button, the selected form will be deleted.

7. After confirming the deletion you will return to the "Form Manager" window where you can select Cancel to close the window.

Pages Script

As described previously, Pages Script is a Basic-like compiled programming language which enables developers to put intelligence behind the forms created in Sanga Pages. Not just for straight-forward data access, Pages Script can be used for form routing, data triggers, validation, process automation, and many other advanced programming applications. Pages Script will be familiar to developers who are experienced with general scripting languages. Furthermore, this chapter is designed for those readers who have some experience with writing shell scripts.

Using Variables

Variable - a variable is a named storage location that holds data that changes; every variable can hold only one kind of data.

Before you can use a variable in Pages Script, you must request that the variable be created by declaring the variable first. To declare a variable means to give the name and data type of the variable. Once you have declared a variable, it always retains its original data type. A Pages Script program has no limit on the number of variables it declares.

Pages Script Data Types

integer	Numeric values with no decimal point or fraction. Integer values range from -32,768 to 32,767.

string	Data that consists of 0 to 2E+38 characters of alphanumeric data. Alphanumeric means that the data can be both alpha betic and numeric. String data values may also contains special characters such as ^%@.

dim Statements - define variables. dim - short for "dimension" - is a Pages Script statement that you write in an application's Script window.

Using dim, you tell Pages Script: that you have a variable, what to name the variable, and what kind of variable you want. Always use a dim statement to define variables before you use variables.

Format of the dim Statement: `dim` *VarName* `as` *DataType*

VarName is a name that you supply. When Pages Script executes the dim statement at runtime, it creates a variable in memory and assigns it the name you give in the VarName location of the statement.
DataType is one of the two Pages Script data types defined in the above table.

For example, the following statement defines a variable named "ProductTotal" :
`dim` *ProductTotal* `as` *integer*
From the dim statement, you know that the variable holds integer data and that its name is ProductTotal.

All Pages Script commands and built-in routines are case insensitive. Although you do not have to name your variables with an initial capital letter, most Pages Script programmers do for the sake of consistency. Additional caps help distinguish individual words inside a name (Remember that you cannot use spaces in the name of a variable).

Note: Never define two variables with the same name. Pages Script won't know which one you are referring to when you try to use one of them.

Shortcut! - When you need to define several variables:

Instead of listing each variable definition on separate lines like this:

```
dim A as integer
dim C as integer
dim D as string
dim E as string
```

You can combine variables of the same data type on one line. For example:

```
dim A, C as integer
dim D, E as string
```

Assigning Values to Variables

null string - a zero-length empty string that is often represented as "".

Defining Variables

Pages Script initializes with zeroes in the numeric variables and null strings in the string variables. Use the "assignment statement" when you want to put other data values into variables.

Format of the assignment statement: *VarName = Expression*

The *VarName* is a variable that you have defined using the dim statement. *Expression* can be a constant, another variable, or a mathematical expression. Example: Suppose you need to store a minimum age value of 18 in an integer variable named MinAge. The following assignment statement does that:
MinAge = 18

Mathematical Expressions

Operator - a word or symbol that performs data manipulation and/or comparison.

Math Operators

Table of Primary Math Operators

Operator	Example	Description
+	Net + Disc.	Adds two values
-	Price - 4.00	Subtracts one value from another
*	Total * Fact	Multiplies two values
/	Tax/Adjust	Divides one value by another

How to Use Mathematical Operators

Suppose that you wanted to store the difference between the annual sales (stored in a variable named AnnualSales) and the cost of sales (stored in a variable named CostOfSales) in a variable named NetSales. Assuming that all three variables have been defined and initialized, the following assignment statement computes the correct value for NetSales:

NetSales = AnnualSales - CostOfSales

This assignment tells Pages Script to compute the value of the expression and to store the result in the variable named NetSales.

No matter how complex the expression is, Pages Script computes the entire result before it stores that result in the variable at the left of the equals sign. In the following assignment statement, for example, is rather lengthy, but Pages Script computes the result and stores the value in a variable named Celsius:

These two lines convert a temperature of 30° Fahrenheit to the correct temperature in Celsius

Fahrenheit = 30
Celsius = (Fahrenheit-32) *9/5

Combining expression often produces unintended results because Pages Script computes mathematical results in a predetermined order. Pages Script then computes all multiplication and division (working from left to right - before any addition and subtraction).

For example:
Pages Script assigns 13 to Result in the following assignment:
Result = 3 + 5 * 2

It is possible to override the operator precedence by using parenthesis. Pages Script always computes the values inside any pair of parentheses before anything else in the expression, even if it means ignoring operator precedence.

Relational Operators

Relational Operators compare data values to one another. They test conditions that are either true or false. And thus they only produce true and false answers. In other words, one data value is either more than another (a true result) or the data value is not more than the other (a false result).

The Relational Operators

Operator	Usage	Description
>	Sales > Goal	The greater than operator. Returns true if the value on the left side of > is numerically or alphabetically greater than the value on the right. Otherwise, false.
<	Pay < 2000.00	The less than operator. Returns true if the value on the left side of < is numerically or alphabeti cally less than the right. Other wise, false.
=	Age = Limit	The equal to operator (some times called the equal operator). Returns true if the values on both sides of = are equal to each other. Otherwise, false.
>=	FirstName >= "Mike"	The greater than or equal to operator. Returns true if the value on the left side of >= is numerically or alphabetically greater than or equal to the value on the right. Otherwise, false.
<=	Num <= lblAmt.Caption	The less than or equal to opera tor. Returns true if the value on the left side of <= is numerically or alphabetically less than or equal to the value on the right. Otherwise, false.
<>	txtAns.Text <> "Yes"	The not equal to operator. Returns true if the value on the left side of <> is numerically or alphabetically unequal to the value on the right. Otherwise, false.

To understand how relational operators work, you must understand how to use their true or false results. The if statement introduced in the next section explains how you can use true and false results to make decisions in your program.

Keep Each Side a Consistent Data Type

The expressions on both sides of a relational operator must have the same data type or at least compatible data types. In other words, you cannot compare a string to a numeric data type. If you try, you will get a Type mismatch error because the data types don't match.

Program Flow Statements

Like all other programming languages, Pages Script supports both conditional statements and loops to determine the program flow. Prior to discussing the flow control structures we should mention Blocks. Blocks consist of one or more Pages Script statements. The block of statements found in a control structure are bounded by the start of the structure and the "ending" statement. For example, an if statement completes with an end if statement, or a do while loop ends with a loop statement. The Pages Script language supports nesting of these program flow structures.

Conditional Statements

The if...then...else Statement

The if statement uses relational operators to test data values. It performs one of two possible code actions, depending on the result of the test. With if statements, Pages Script tests whether to execute blocks of code. The if statement makes decisions. If a relational test is true, the body of the if statement executes.

One format of if:

```
if relationalTest then
      Block
end if
```

The **end if** statement informs Pages Script where the body of the if statement ends. For example;

```
if state = "New York" then
        TaxRate = .0775
end if

if state = "Massachusetts" then
        TaxRate = .06
end if
```

The body of the if executes only if the relational test is true. Otherwise, the rest of the program continues as usual.

A Shortcut Form of If called the Single-Line Format : if relationalTest then PSStatement.

This format does not require an **end if** statement because the relational test and body of the if are all in one line.

Handling False Conditions

The **else** statement executes code based on the relational test's false condition. The else statement is part of an extended if statement that specifies the code that executes if the relational test is false.

Format of If Statement with **else**:

```
if relationalTest then
        Block
else
        Block
end if
```

For example:

```
if (BankBalance < 0) then
        MessageBox ("Overdrawn")
else
        MessageBox ("In Credit")
end if
```

If the condition (BankBalance < 0) is true, then Overdrawn is displayed; if it is not, then In Credit is displayed.

Loop Statements

Loops are used when the same set of steps have to be carried out many times.

The do while... loop

The do while statement performs flow control based on a relational expressions just as the if statement does. The relational expression controls the looping statements rather than a single block of code. Many lines of your program will still execute sequentially, but a loop will cause blocks of code to repeat zero, one or many times.

Format of the do while... loop:

```
Do While (relational test)
        Block
Loop
```

The block of code continues looping as long as the relational test is true. You can put as many lines of code in the block as long as the block itself somehow changes a variable used in the relational test. The block of code will keep repeating as long as the do while loop's relational test continues to stay true. Eventually, the relational test must become false - otherwise you will encounter an infinite loop.

An example of the use of the do while... loop:

```
answer =""
Do while answer <> "N" && answer <> "Y"
  Answer = toUpper(InputBox("Do you want to delete this record?"))
loop
```

The for... loop

The for loop is sometimes called the "For-Next" loop. Unlike the do loops, the for loop repeats for a specified number of times. A for loop always begins with the for Statement and ends with the next statement

Format of the for... loop:

```
for CounterVar = StartVal to EndVal [step IncrementalVal]
        Block
next CounterVal
```

The loop in the following example computes the total of the numbers from 1 to 10.

```
Sum = 0
for Number = 1 to 10
        Sum = Sum + Number
next Number
```

The Exit For Statement

Sometimes, you'll be processing user input or several data values using looping statements, and an exception occurs in the data that requires and immediate termination of the loop. For example, you may be collecting sales values for a company's 10 sales divisions inside a For loop that iterates ten times. However, the user can enter zero for a division's sales value, indicating that there is no sales data for that division. Rather than complete the loop, your program might need to quit the loop at that point because the full divisional report information can't be gathered at the time.

The exit for statement automatically terminates the loop. No matter how many iterations are left in the for loop, when Pages Script encounters an exit for statement, Pages Script immediately quits the loop and sends execution down to the statement following the loop.

Typically, an if statement triggers one of the exit statements like this:

```
for Divisions = 1 to 10
        Block                   // Code to get a sales value
        if (sales = 0) then
            exit for            // Quit the loop early
end if
        Block
next Divisions
```

The if ensures that the exit for executes only under one specific condition (a missing sales value). Without that specific condition triggering the exit for, the loop cycles normally. The exit for construct can only be used from within a for loop and is a way of leaving the loop immediately.

Utility Functions

There are about 70 utility functions that are built into Sanga Script. These provide useful shortcuts and functionality to the Script user.

To use these functions you must type the name in exactly as shown below although the functions are case insensitive. These functions can be broken up into about 5 different groups Database, Screen, Math, String, Other.

Screen Functions

The Screen functions manipulate the different things you see on the screen.

MessageBox(msg) returns void.
where msg is the message you wish to display.

This function puts a box up on the screen with the contents of msg inside. The script will then WAIT FOR A MOUSE CLICK before proceeding. The argument msg may be a string or a number.

ConfirmMessageBox(msg) returns an integer.
where msg is the message you wish to display.

This function puts a box up on the screen with the contents of msg inside. Additionally, there will be two buttons with YES and NO inside the box. The user must click on one for the script to proceed. The function will return a 1 if the user clicks on YES and 0 if the user clicked on NO.

ScreenField(fid) returns String
where fid is the legitimate ID of an object on the screen.

This function returns the contents of the fid object. If the fid object is a list or a choice, it will return the chosen element. If there is no answer, the function will return an empty string.

setScreenField(fld, newValue) returns void
where fld is the legitimate ID of an object on the screen.
where newValue is what value you wish fld to take on.

This function sets the contents of the fld object to the contents of the newValue variable. To set the list or choice objects, you must use setListElement.

hideField(fld), showField(fld) both return void
where fld is the legitimate ID of an object on the screen.

These affect the requested object on the screen. The hideField function makes the requested object invisible and disabled. The user cannot see nor act on the screen object. The showField function makes the object visible and enabled.

disableField(fld), enableField(fld) both return void
where fld is the legitimate ID of an object on the screen.

These affect the requested object on the screen. The disableField function makes the requested object gray and dead. The user cannot act upon the screen object. The enableField function redraws the object and makes it work again.

fieldIsSet(fld), fieldIsReset(fld) both return int
where fld is the legitimate ID of an object on the screen.

These are only to be used on checkboxes and radio buttons. These two screen element types can be in 1 of 2 states. Either they are ON or OFF. If the requested screen object is ON, then the function returns 1, if the requested screen object is OFF, then the function returns 0.

setField(fld), resetField(fld), toggleField(fld) all return void
where fld is the legitimate ID of an object on the screen.

These are only to be used on checkboxes and radio buttons. These two screen element types can be in 1 of 2 states. Either they are on or off. The setField turns the requested object ON. The resetField turns the requested object OFF. The toggleField object changes the state from ON to OFF or OFF to ON as the case requires. An object that is part of a group may not be turned off. You must turn ON a different object in that group.

clearList(fld) returns void
where fld is the legitimate ID of an object on the screen.

This function is only to be used on a choice or list screen object. This function takes all the elements or selections off of the desired object.

addListEntry(fld, entry), removeListEntry(fld, entry) returns void
where fld is the legitimate ID of an object on the screen.
where entry is the String representation of the element to be affected

These functions are only to be used on a choice or list screen object. The addListEntry function puts the String variable entry onto the end of the choice or list object. This new entry may now be chosen by the user. The removeListEntry takes the specified entry off the list.

setMultipleSelections(fld),resetMultipleSelections(fld) return void
where fld is the legitimate ID of an object on the screen.

These functions should only be used on a list screen object. A list screen object may be used to make multiple selections off the list. You must first tell the list object to work in this manner. A list screen object will only allow one selection by default. Calling the setMultipleSelection function with the fld argument as the screen ID will make the list screen object allow multiple selections. Calling the resetMultipleSelection function with the fld argument as the screen ID will make the list screen object not allow multiple selections.

getNumberSelected(fld) returns int
where fld is the legitimate ID of an object on the screen.

This function should only be used on a list screen object. This function will return the number of entries the user has selected. This number may be anywhere from 0 to the total number of entries on the list.

getSelection(fld, selectionNumber) returns int
where fld is the legitimate ID of an object on the screen.
where selectionNumber tells which selection you wish returned

This function should only be used on a list screen object. Once you know the number of selections the user has chosen via the getNumberSelected call, this function can be used to get each individual selection. If there are more than one entries selected, then you must make multiple calls to this function.

Database functions

DataBaseField(fld) returns String
where fld is the legitimate ID of a field in the database.

This function returns the contents of the field in the current record of the database. If there is no such field, or no current record, then the function will return an empty string.

setDataBaseField(fld, newValue) returns void
where fld is the legitimate ID of an object on the screen.
where newValue is what value you wish fld to take on.

This function sets the contents of the field in the current record of the database to the contents of the newValue variable. If there is no such field, then the function will do nothing.

update() returns void

This function copies the contents of the screen to the database to be saved.

delete() returns void

This function will remove the current record from the database.

addnew() returns void

This function will get a blank record ready to be added to the database. You must follow this call with a call to update if you wish to permanently add this function to the database.

movenext() returns void

This function changes the current record of the database by moving the record after the current record. All screen fields will automatically be changed to reflect this.

movebrev() returns void

This function changes the current record of the database by moving the record before the current record. All screen fields will automatically be changed to reflect this.

movefirst() returns void

This function changes the current record of the database by moving the record to the very first record. All screen fields will automatically be changed to reflect this.

movelast() returns void

This function changes the current record of the database by moving the record to the very end record. All screen fields will automatically be changed to reflect this.

moveto(recNumber) returns void
where recNumber is the number of the new record you wish to go to

This function changes the current record of the database by moving the record to the desired position in the database which is specified by the recNumber variable. All screen fields will automatically be changed to reflect this.

getcurrecnum() returns int

This function returns a number which represents your position in the database. This number will always be positive unless you are in the midst of adding a new record to the database.

String Functions

strlen(str) returns int

This function returns a count of the number of characters in the variable str.

right(str, count) returns String

This function returns a new string which is identical to the count rightmost characters of the variable str.

left(str, count) returns String

This function returns a new string which is identical to the count leftmost characters of the variable str.

midstr(str, start, count) returns String

This function returns a new string which is identical to the count characters starting at the start character of the variable str.

concat(str1, str2) returns String

This function returns a new string which is identical to with str2 added on to the end of str1.

ntos(number) returns String

This function returns a new string which is the representation of number in string form.

ston(str) returns Integer

This function returns a number which is the equivalent of the string representation.

strcmp(str1, str2) returns Integer

This function returns a number which is computed by comparing the string in str1 with the string in str2. If str1 is the same as str2, the function returns a 0. If str1 is alphabetically earlier than str2, then the function returns a -1. Otherwise, the function returns a 1.

56 Pages

stricmp(str, count) returns String

This function is the same as strcmp above, with the exception that an uppercase letter is considered to be equal to the same lowercase letter.

toupper(str), tolower(str) return String

This function returns a new string which is created by taking the argument and converting any lowercase characters to uppercase. Or uppercase to lower for the tolower function.

trim(str) returns String

This function returns a new string which is created by taking the argument and removing any leading or trailing spaces.

charAt(str, position) returns String

This function returns an integer representing the character at a specified position in the string.

indexofchr(str, character) returns integer

This function computes and returns the position in the string of the character argument. If the character is NOT in the string, then the function returns -1.

indexofstr(str1, searchString) returns integer

This function computes and returns the position in the string of the second string argument. If the search string is NOT in the string, then the function returns -1.

nextindexofchr(str, character, startPosition) returns integer

This function is identical to the indexofchr function with the exception that the searching does not start with the first character but instead will start searching at a position in the third argument.

nextindexofstr(str, character) returns integer

This function is identical to the indexofstr function with the exception that the searching does not start with the first character but instead will start searching at a position in the third argument.

Math Functions

abs(number) returns Floating Point Number

This function computes the absolute value of the argument. That is, it makes the number positive if it is not already so, and then returns it.

ceil(number) returns Integer Number

This function computes and returns the smallest whole number that is larger than the argument.

floor(number) returns Integer

This function computes and returns the largest whole number that is smaller than the argument.

log(number) returns Floating Point Number

This function computes and returns the natural log of the argument.

sqrt(number) returns Floating Point Number

This function computes and returns the square root of the argument.

pow(number, power) returns Floating Point Number

This function computes and returns the result of the first number raised to the power of the second number.

max(number1, number2) returns Floating Point Number

This function computes and returns the larger of the two arguments.

min(number1, number2) returns Floating Point Number

This function computes and returns the smaller of the two arguments.

Miscellaneous Functions

copytoscreen(void) return void

This function copies all database fields into their corresponding fields on the screen.

copytobook(void) return void

This function copies all screen fields into their corresponding fields in the database.

getdate(void) return String

This function computes the current date and returns it as a string. The date is in MM/DD/YY format.

gettime(void) return String

This function computes the current local time and returns it as a string. The time is in 24 hour HH:MM:SS format.

Pages Security

All Client and Service sessions are encrypted using DES (Data Encryption Standard) encryption. The Pages Server administrator is also able to control which users and/or groups have access to the data sources and what kind of access the user/group will have. For further information on administrating an ACL (Access Control List) of the data source, please refer to the chapter on Pages Server.

60 Pages Security

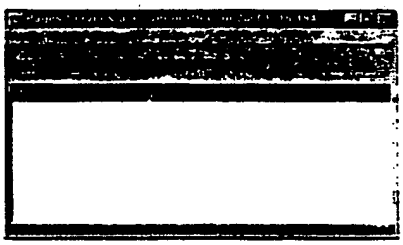
Pages Server

To start the Pages Server:

1. Open System Command Shell (DOS shell in win95, c-shell in UNIX, et cetera)
2. Change to the installation directory of Sanga Pages.
3. Invoke the "Start Server" shell script by entering the following:

```
StartPagesServer
```

4. A new window will appear called "Pages Server hostname/IP address" This window's menu bar has three options: Action, Admin, and Options. The onlyother features of this window is a list where system messages generated andreceived by this service are displayed. Once you have started Pages Server, the message "The Server is running..." should be highlighted to indi cate that it is the most recent system message.



Action

Starting the Server

1. In the "Pages Server" window, select Start from the Action dropdown menu.
2. A message will appear in the System Messages list: "The Server is running..."

Stopping the Server

1. In the "Pages Server" window, select Stop from the Action dropdown menu.
2. A message will appear in the System Messages list: "The Server is stopped".

Starting the Web Server

1. Select Start the Web Server from the Action dropdown menu.
2. A message will appear in the System Messages list: "The Web Server is running..."

Stopping the Web Server

1. Select Stop the Web Server from the Action dropdown menu.
2. A message will appear in the System Messages list: "The Web Server is stopped."

Exiting Pages Server

1. Select Exit from the Action dropdown menu to shut down and exit from Pages Server.
2. Close the Command Prompt window after Pages Server is stopped.

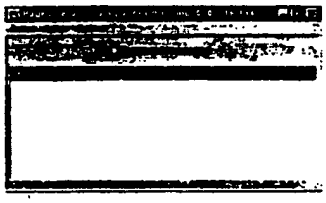
Admin

Login

Login is the only option available in the Admin dropdown menu when you first start Pages Server. Once you have logged in properly, the other options will become active.

1) Select Login from the Admin dropdown menu.

2) The "Pages Root Login" window appears on the screen. Enter the Pages Root password at the text prompt.



3) Once you have entered the password (it will appear in asterisk form), select OK to continue logging into the service or Cancel to return to the "Pages Server" window without logging in.

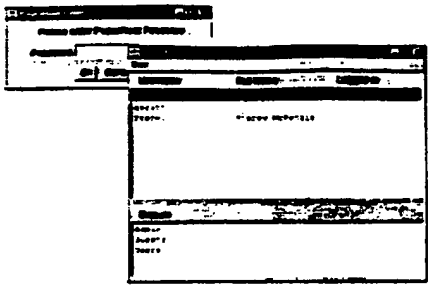
4) If you selected OK and the password was validated, you will return to the "Pages Service" window. Login should now have a checkmark in front of it to indicate that you have logged on successfully. The other options of the "Pages Server" menubar will now be available. If there was a problem with the login (an invalid password perhaps), then an error message will be displayed in the System Messages list and you will be prompted to try again.

User Manager

User Manager acts as an administrative tool for managing users and groups on the service.

1) Select User Manager from the Admin dropdown menu.

2) A new window appears called "User Manager". All the users on the service are listed by the Username, Full Name, and the Login status. All the groups on the service are also listed.

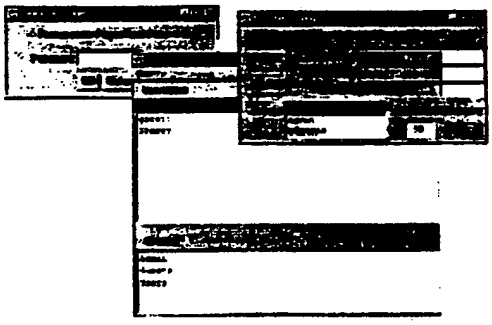


3) The User dropdown menu has five options: Add User..., Add Group..., Delete, Properties, and Exit. Instructions for using each of these options to follow.

Adding a User

1) Select Add User... from the User dropdown menu in the "User Manager" window.

2) A new window appears called "Add User Dialog".



3) To add another Pages user, enter the user name in the text box following the prompt for User Name. It must be at least three characters in length and should not contain any spaces.

4) Enter the real name of the user in the text box following the prompt for **Real Name**.

5) Enter the password for the user in the text box following the prompt for **Password**.

6) Select which **Groups** you want this user to belong to. The user must belong to at least one group.

7) You can also set the date when you want the password to expire by entering in the number of days in the box provided. The number must be between 0 and 90.

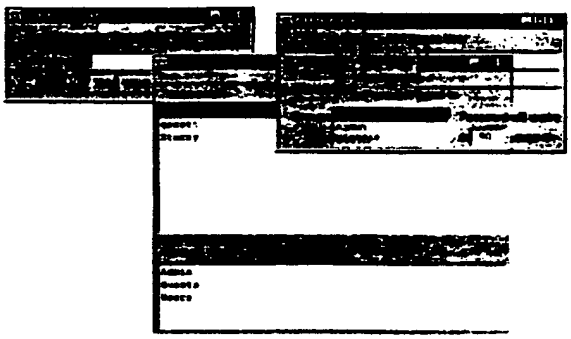
8) Select **OK** if you want to save the information you have entered about the user. You will return to the "User Manager" window where the new user should now appear in the User list.

9) Select **Cancel** to return to the "User Manager" window without saving the entered information.

Adding a Group

1) Select **Add Group...** from the **User** dropdown menu in the "User Manager" window.

2) A new window appears called "Add Local Group".



3) Enter the new group name in the text box following the prompt for **Group Name**.

4) Now you can add to or remove Local Users from the Group Members list

5) Select OK to save the information you have entered about the group. You will return to the "User Manager" window where the new group should now appear in the Group List.

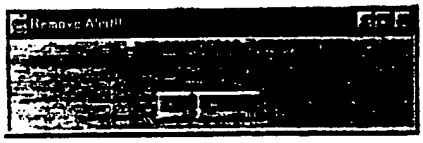
6) Select Cancel to return to the "User Manager" window without saving the entered information.

Deleting Users or Groups from Pages Server

1) In the "User Manager" window, select the User or Group that you want to delete from the appropriate list.

2) Select Delete from the User dropdown menu.

3) A "Remove Alert!!" window appears to tell you that your action will remove the selected User/Group.



4) Select OK to complete the deletion of the User or Group. You will return to the "User Manager" window and you should notice that the User/Group is no longer in the appropriate list.

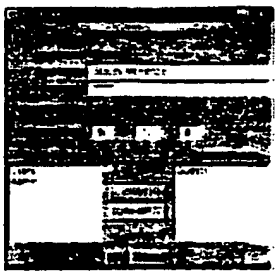
5) Select Cancel to abort the removal of the User/Group from Pages Servers. You will return to the "User Manager" window.

Administrating the Properties of Users

1) Select the User you wish to work on from the Users list.

2) Select Properties from the User dropdown menu.

3) A new window appears entitled "User Properties" as illustrated below



4) As you can see in the diagram, **User Name** cannot be changed. However, you can change the real name of the user by entering it in the text box following the prompt for **Real Name**.

5) To change the password for the user, enter the new password in the text box following the **Password** prompt. The password will, of course, appear in asterisk form.

6) To change the expiration date for the password, simply enter in the new date in the text boxes provided. The date must be between the current date and December 31, 1999 (99/12/31).

7) To add or remove the user from **User's Groups** or **System Groups**, use the tools at the bottom of the window. Clicking on a group in the list of **User's Groups** and then selecting **Remove->** will remove the selected group from that list and place it in the **System Groups** list. Clicking on a group in the list of **System Groups** and then selecting **Add->** will remove the selected group from that list and place it in the **User's Groups** list. Remember that a user must belong to at least one group so the **User's Groups** list cannot be empty.

8) Select **OK** to save your changes. You will return to the "User Manager" window.

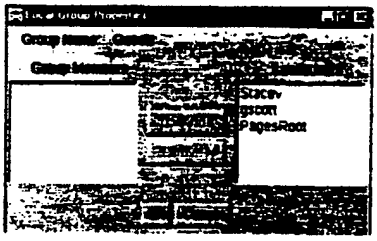
9) Select **Cancel** to return to the "User Manager" window without saving any changes.

Administrating the Properties of Groups

1) In the "User Manager" window, select the Group whose Properties you wish to view.

2) Select Properties from the User dropdown menu.

3) A new window appears entitled "Local Group Properties" (as illustrated below)



4) As you can see in the diagram, this window allows you to add and remove Local Users to the Group where they become Group Members. Clicking on a group in the list of Groups Members and then selecting Remove-> will remove the selected group from that list and place it in the Local Users list. Clicking on a group in the list of Local Users and then selecting Add-> will remove the selected group from that list and place it in the Groups Members list.

5) Select OK to save the changes that you have made and to return to the "User Manager" window.

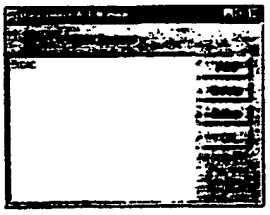
6) Select Cancel to return to the "User Manager" window without saving any changes you have made.

Exiting the "User Manager" window

Select Exit from the User dropdown menu to close the "User Manager" window and return to the "Pages Service" window.

Data Source ACL Manager

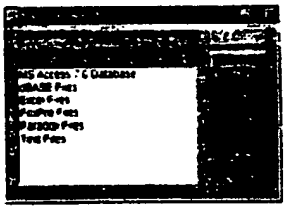
The Data Source ACL Manager provides a way to attach an Access Control List (ACL) to the data sources provided by the service. These lists are used to ensure the security of the data sources and to provide access only to authorized users. To use the Data Source ACL Manager, select **Data Source ACL Manager** from the **Admin** dropdown menu. The "Data Source ACL Manager" window appears, displaying a list of configured data sources and various actions that can be performed (**Add**, **Delete**, **Setup**, and **OK**).



Data Source List: This is a list of data sources for which an ACL has previously been configured. Selecting an item in this list will allow you to either **Delete** or **Setup** the data source ACL. Both of these options are described below.

Adding a Data Source ACL

1. Select the **Add** button in the "Data Source ACL Manager" window.
2. A new window appears called "Add Data Source ACL". This window displays a list of all the data sources provided by the service.



3. Select the data source for which you want to add an ACL and click **OK**, or
4. Select **Cancel** to return to the "Data Source ACL Manager" window without creating a new ACL.

5. If you selected OK in step 3, then the "Data Source ACL Setup" window will appear so you can start configuring a new ACL for the data source. Please refer to the section on Setting Up a Data Source ACL for information on this window.

Deleting Data Sources from the ACL

1. Select the data source from the list displayed in the "Data Source ACL Manager" window.

2. Click on the **Delete** button.

3. A "Remove Alert!!" window will appear to warning you that you are about to remove the data source ACL.

4. Click **OK** to continue with the deletion and to return to the "Data Source ACL Manager" window where you will notice the data source is no longer present in the list.

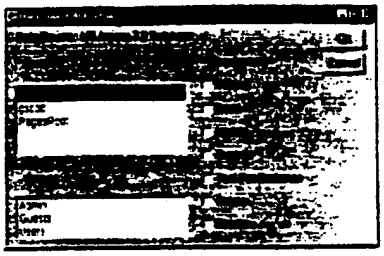
5. Click **Cancel** to stop the deletion process. You will return to the "Data Source ACL Manager" window.

Setting Up a Data Source ACL

The configuration of a data source ACL is managed by the "Data Source ACL Setup" window. This window can be entered by either:

A) Adding a new data source and then setting the initial configuration (described in the previous section). In this case the blank ACL will be established with no permissions set.

B) Select a data source from the list in the "Data Source ACL Manager" window and clicking on the Setup button. When entering this way the ACL's previous values will be loaded.



The above illustration displays the Data Source ACL Setup window which contains the following areas:

1. The Name of the Data Source is displayed at the top. This value cannot be changed as it must match the actual data source value.

2. Below this are two lists, a Users list and a Groups list. The Users list contains all local users managed by this service. The Groups list contains the default system wide groups as well as any locally defined groups.

3. Next to these lists are the Pages Permissions which are made up of the following set:

   - Control - for the creator of the data source.
   - Read - allows read-only access to the data source.
   - Update - allows the user/group to update any changes made to the data source.
   - Execute - allow the user/group to execute a file outside of Sanga Pages.
   - Test for Existence - determines whether or not the user has access to see data source.
   - Delete - allows the User/Group access to delete a data source, and
   - Create - allow the User/Group access to create a data source.

Whenever a user or group is selected these check boxes will reflect the permission set for that user/group. To add (box contains a ✓) or remove (box is empty) a permission simply click on the appropriate field and the value will be toggled.

4. Select OK to save the ACL and return to the "Data Source ACL Manager" window. If this was a new data source ACL being created then the data source should now appear in the list of data sources.

5. Select Cancel to ignore all changes to the data source ACL and return to the "Data Source ACL Manager" window. If this was a previously existing data source ACL then the original ACL will still be intact. If this was a new ACL then all entered data will be lost and no ACL will be created for the data source.

Exiting the "Data Source ACL Manager" window.

Selecting the OK button will close the "Data Source ACL Manager" window and return you to the main "Pages Service" window.

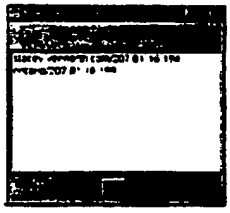
Options

The third option available in the "Pages Server" window is Options which has three additional tools for the Pages Server administrator.

Show Services

In the "Pages Server" window, select Show Services from the Options dropdown menu.

A new window appears called "Available Services". A list of all the Pages Servers available on your intranet is displayed.

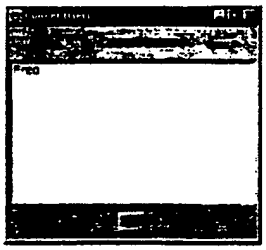


Click OK to return to the "Pages Server" window.

Current Users

1) In the "Pages Server" window, select Current Users from the Options dropdown menu.

2) A new window appears called "Current Users". A list of all the users currently logged on the server are displayed.



3) Click OK to return to the "Pages Server" window.

Broadcast Message

The Broadcast Message tool is useful for informing other users on the Server about any changes happening on the Service (i.e. "My server will be stopped in five minutes") or perhaps any social events happening in the company (i.e. "Big party at the FireStation Friday night at 8 p.m.").

1) In the Pages Server window, select Broadcast Message from the Options dropdown menu.

2) A new window appears called "Broadcast Message".(insert options3.bmp)

3) Enter the message you wish to send in the text box following the **Message** prompt.

4) Click **OK** to send the message and to return to the "Pages Services" window, or

5) Click **Cancel** to exit the window without sending a message. You will return to the "Pages Services window".

The foregoing discussion discloses and describes merely exemplary methods and embodiments of the present invention. As will be understood by those familiar with the art, the invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.

CLAIMS

What is claimed is:

1. A system for accessing data stored in a plurality of storage mediums and in a plurality of storage formats, comprising:

a client module having browserware application properties and providing a desktop having a plurality of books each associated with data stored in a storage medium; and

a server module operatively coupled to the client module, the server module including computer instructions for performing the method comprising:

storing information associating the book with metadata for retrieving the data associated with the book;

receiving an access request identifying a data source from the client module;

retrieving metadata corresponding to the data source;

retrieving data corresponding to the metadata; and

sending the retrieved data to the client module.

2. The system of claim 1 wherein the client module runs in a browser environment.

3. The system of claim 1 wherein the client module is a platform independent module.

4. The system of claim 1 further comprising a scripting module having a byte-code generator disposed to generate byte-code from a form layout.

5. The system of claim 1 wherein the storage medium arranges data in a row and column format and wherein the book comprises:

a data source identifier;

a plurality of attributes each identifying a column;

a join condition; and

view information.

6. The system of claim 5 further comprising a result object for returning the result of a data source query generated from the data source identifier, the plurality of table identifiers and the join condition.

7. The system of claim 6 wherein the result object is presented in accordance with the view information.

8. The system of claim 5 wherein the view information includes a selection of data attributes and a search condition.

9. The system of claim 5 wherein the view information further includes a plurality of forms.

10. A computer implemented method for accessing data stored in a plurality of storage mediums and in a plurality of storage formats, comprising:

providing a client module having browserware application properties and having a desktop having a book identifying data stored in a storage medium;

providing a server module storing information associating the books with metadata for retrieving the data identified by the book;

receiving at the server module a data access request from the client module;

retrieving metadata responsive to the access request; and

retrieving data stored in accordance with the metadata;

sending the retrieved data from the server module to the client module.

11. The method of claim 10 wherein the client module runs in a browser environment.

12. The method of claim 10 wherein the client module is a platform independent module.

13. The method of claim 10 further comprising:

providing a form layout interface having user customizable data display attributes;

processing the customized form to generate byte-code disposed to display retrieved data in accordance with the customized form.

14. The method of claim 10 further comprising:

providing a user interface for identifying a data source;

determining the physical storage information associated with the identified data source;

generating metadata from the determined physical storage information; and

storing the metadata.

15. The method of claim 14 wherein the identified data source identifies data stored in first and second tables in a database and includes a table join condition.

16. The method of claim 14 wherein the identified data source is an electronic mail system.

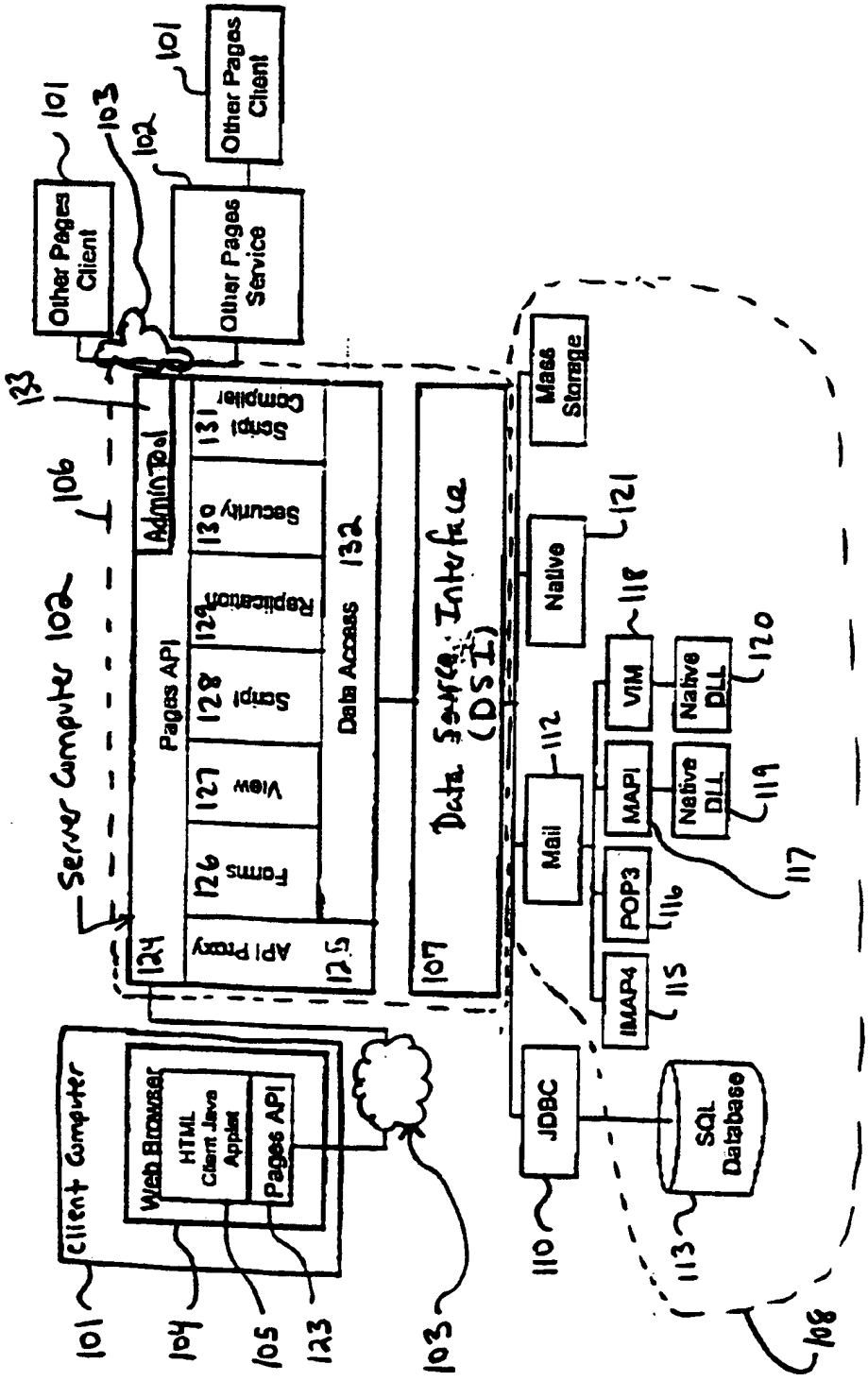

Other Pages Client
Other Pages Client
Other Pages Service
Server Computer
AdminTool
Script Compiler
Security
Replication
Script
View
Forms
Pages API
API Proxy
Data Access
Data Source Interface (DSI)
Mass Storage
Native
VIM
Native DLL
MAPI
Native DLL
Mail
POP3
IMAP4
JDBC
SQL Database
Client Computer
Web Browser
HTML Client Java Applet
Pages API

Figure 1

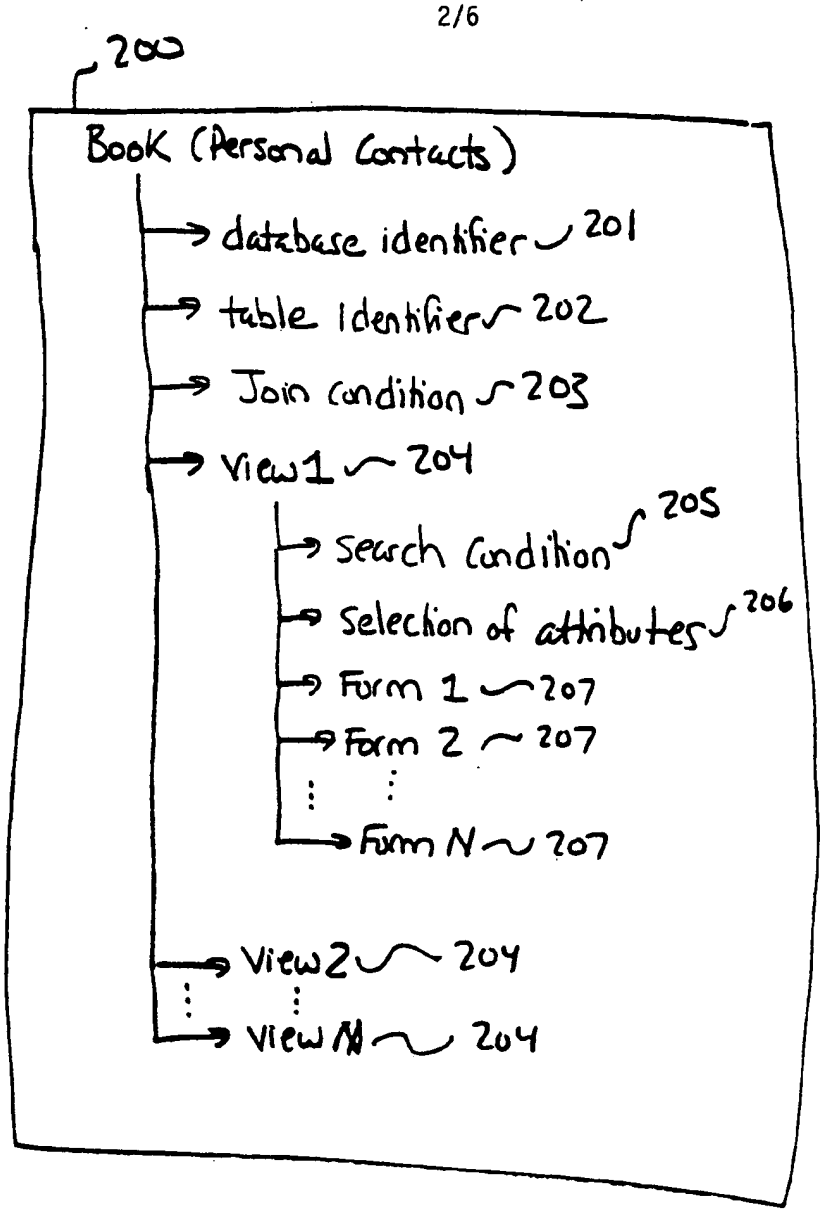

200
Book (Personal Contacts)
database identifier 201
table Identifier 202
Join condition 203
View1 204
Search Condition 205
Selection of attributes 206
Form 1 207
Form 2 207
Form N 207
View2 204
View N 204

Figure 2

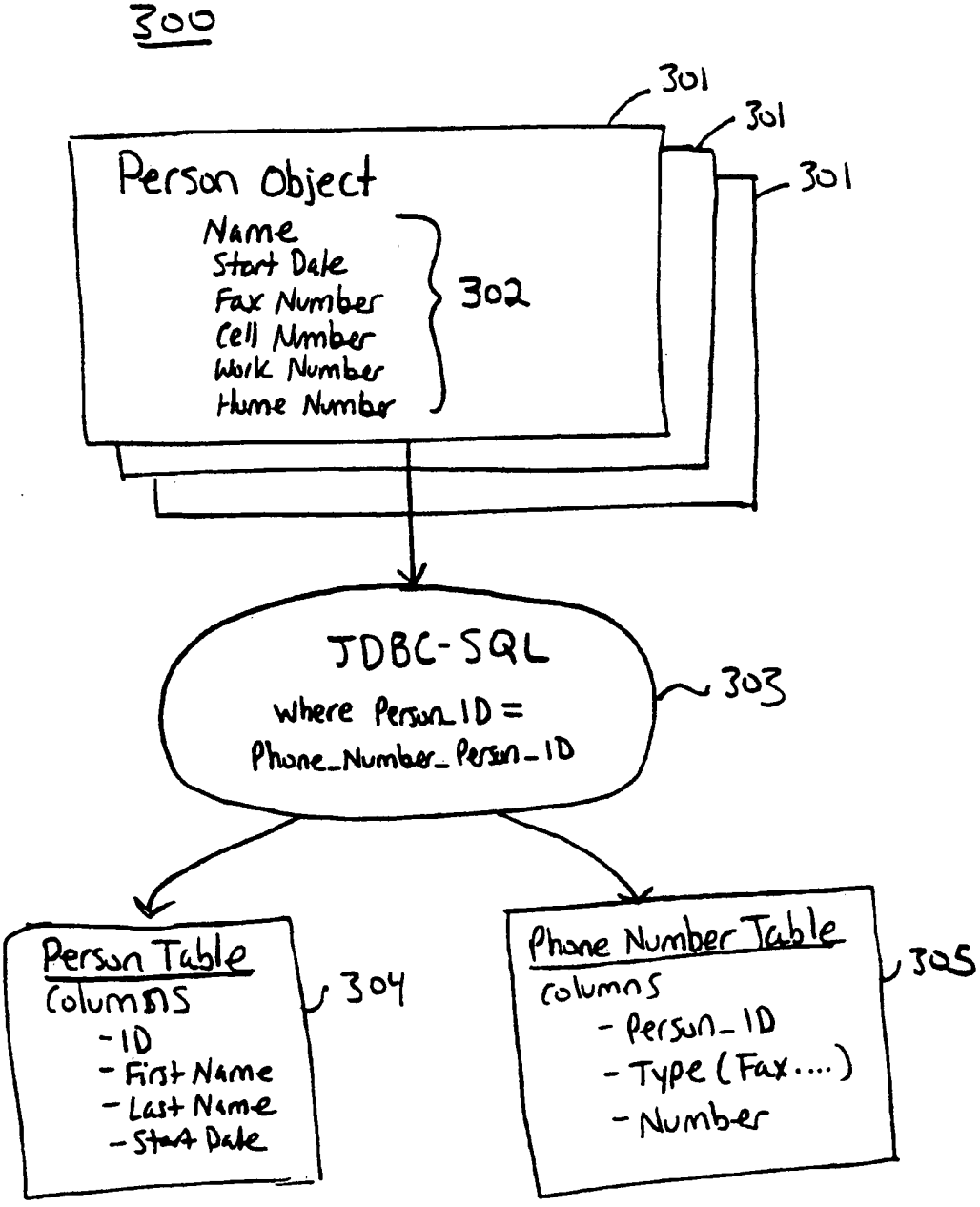

300
301
301
301
Person Object
Name
Start Date
Fax Number
Cell Number
Work Number
Home Number
302
JDBC-SQL
where Person_ID =
Phone_Number_Person_ID
303
Person Table
Columns
- ID
- First Name
- Last Name
- Start Date
304
Phone Number Table
columns
- Person_ID
- Type (Fax....)
- Number
305

Figure 3

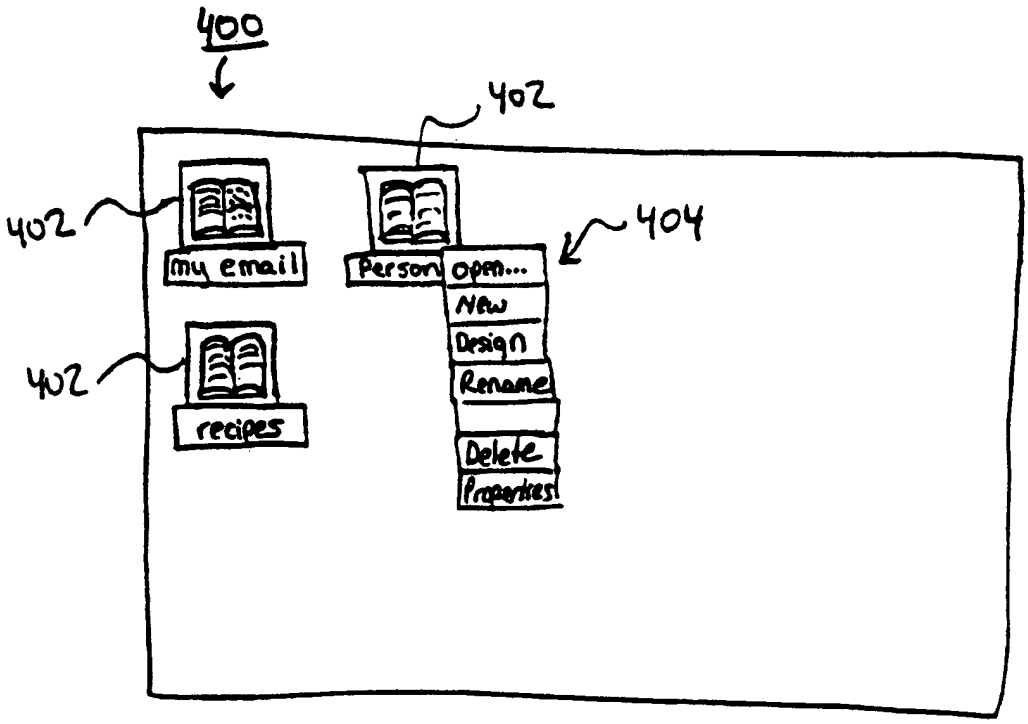

400
402
402
my email
402
recipes
Person
404
Open...
New
Design
Rename
Delete
Properties

Figure 4

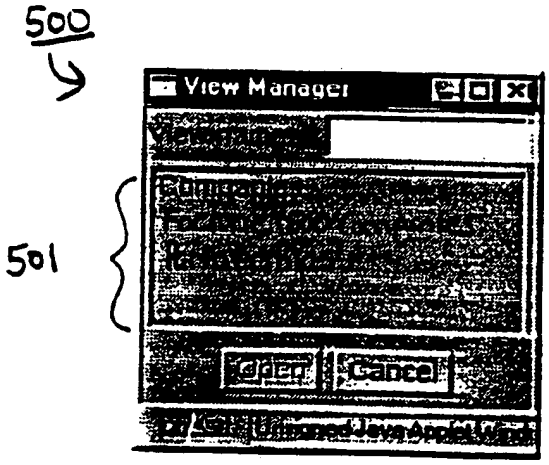


Figure 5

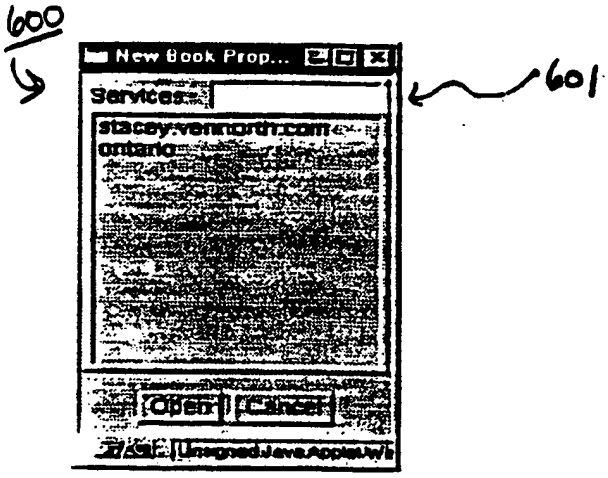


Figure 6A

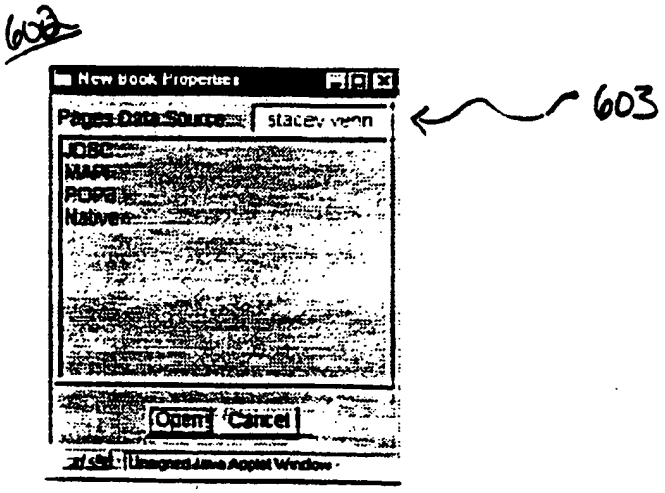


Figure 6B

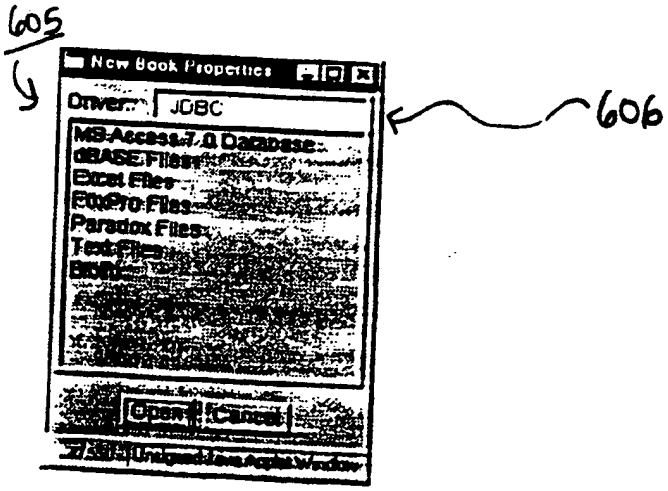


Figure 6C